# CHARACTERIZATION OF LIBRARY CELLS FOR OPEN AND SHORT CIRCUIT DEFECT EXPOSURE: A SYSTEMATIC METHODOLOGY

A Thesis
Presented to
The Academic Faculty

By

Sanya Gupta

In Partial Fulfillment
of the Requirements for the Degree
Master of Science in the
School of Electrical and Computer Engineering

Georgia Institute of Technology

December 2019

**CHARACTERIZATION OF LIBRARY CELLS FOR OPEN AND SHORT CIRCUIT DEFECT EXPOSURE: A SYSTEMATIC METHODOLOGY**

Approved by:

Dr. Abhijit Chatterjee, Advisor
School of Electrical and Computer Engineering
*Georgia Institute of Technology*

Dr. Saibal Mukhopadhyay
School of Electrical and Computer Engineering
*Georgia Institute of Technology*

Dr. David C Keezer
School of Electrical and Computer Engineering
*Georgia Institute of Technology*

Date Approved: December 6, 2019

Education would be much more effective if its purpose was to ensure that by the time they leave school every boy and girl should know how much they do not know, and be imbued with a lifelong desire to know it.

*William Haley*

## ACKNOWLEDGEMENTS

Undertaking this research as my Master's thesis has been a life-changing experience for me. It would not have been possible without the support and guidance of many people.

I would like to express my sincere gratitude towards my advisor, Dr. Abhijit Chatterjee who has inspired, supported and guided me with his immense knowledge in the field of Very Large Scale Integration system testing. Without his kind support, encouragement, and feedback, this thesis would not have been achievable.

I would like to thank Suriya Natarajan and Arani Sinha from Intel Corporation for the guidance and inputs provided by them towards making this thesis industrially relevant and an improvement over current technology. Also, I gratefully acknowledge the funding received towards my Master's thesis from Semiconductor Research Corporation under GRC Task 2901.01.

This work was performed in collaboration with my research partner Sujay Pandey (Ph.D., ECE, Georgia Tech), who has played an instrumental role in defining the path and the attainment of the goals of this research. I would also like to express my gratitude towards my labmates and friends at SRSL, Georgia Tech for their constant support. I would like to appreciate the help provided by the ECE Graduate Office at Georgia Tech in all the administrative matters.

Last but not least, I would like to thank my parents, family and friends for their every lasting support and motivation which helped me stay positive during the entire course of my degree and thesis.

## TABLE OF CONTENTS

## LIST OF TABLES

## LIST OF FIGURES

## SUMMARY

This thesis aims to systematically develop a test generation methodology for the detection and characterization of intra-cell manufacturing defects occurring at transistor level in standard library cells, which are used extensively as building blocks of an integrated circuit chip. In this work, we focus on developing test vector generation algorithms for the intra-cell resistive open and short circuit defects. For this, we performed an exhaustive set of simulations for different defect locations with varying magnitudes of the resistive defect, to demonstrate the requirement for multi-bit change and multi time frame patterns for improving the fault coverage over traditional testing methodologies. Based on the observations from the exhaustive simulation and the principles of device physics we develop a test generation algorithm for the detection of these defects and we also present the correlation between the coverage attained through the algorithm generated patterns and the coverage attained through exhaustive simulation. We also demonstrate that there is a significant amount of speed-up obtained due to the guided pattern simulation using the algorithmic approach, in the detection of these defects.

# CHAPTER 1
# INTRODUCTION

One of the major challenges of advancement in technology scaling and voltage scaling is an increased number of defects escaping the traditional testing methodologies. Structural test methods based on fault models such as stuck-at, transition delay, and path delay are deployed using Automatic Test Equipment (ATE) on the hardware of a circuit for the detection of defects. Structural tests are preferred over system-level tests for a complete manufactured chip due to the cost involved with it. However, the technology scaling has led to the loss of defect coverage by using the traditional structural test which necessitates the need for expensive system-level testing, which subsequently increases the cost of testing each Integrated Circuit (IC) chip. This is because traditional fault models mostly consider defects at the cell inputs, outputs and the interconnects between them, but with technology scaling a significant number of defects occur within and between instances of reusable, building blocks of a circuit known as standard cells. To reduce the cost of testing by improving the structural testing methodology, a significant amount of research involving advance fault modeling and detection of physical defects in manufactured ICs has been performed [1], [2], [3].

With the intent to minimize Defective Parts Per Million (DPPM) in manufactured chips [4], Cell-Aware Test (CAT) methodology targets the defects within standard cells (intra-cell defects) by injecting parasitic extracted netlist of each cell in the cell library with open and short circuit defects and using the cell input stimulus detecting these defect to generate scan test pattern to expose such defects in netlist containing the instances of these cells [5]. CAT methodology has proven to be effective in the reduction of DPPM over traditional stuck-at and transition 5-detect tests for experiments performed on an AMD 32nm notebook processor, as demonstrated in [5]. Unfortunately, even after screening through CAT, hazard

activated open defects can cause failure at system level test, as shown in [6]. [7] describes the challenges associated with PVT (Process, Voltage, and Temperature) variation, faced by CAT methodology. In [8], another approach of test pattern generation based on logical function is described.

In this research, we focus on open and short-circuit defects within the cells. Due to a significant amount of simulation overhead, prior methodologies including CAT methodology, characterize each resistive defect in each standard cell with one size of extreme magnitude. Generally, open defects are characterized by a large resistance value and short defects are characterized by a small resistance value. This approach does not account for the behavior of defects with different magnitude. Also, prior methodologies only perform up to two time frame simulation tests with limited test patterns having a single bit changing in the two vectors.

In this work, we demonstrate using exhaustive simulation over a set of magnitudes and locations for open and short-circuit defects, that defects of certain magnitude may remain undetected by previous test methodologies and will require either multi-bit switching input pattern, multi time frame input pattern or a combination of both, for their detection. Results from the exhaustive simulation are used to gain insight to construct test generation algorithms based on switch level, Elmore delay model, for guided simulation approach which reduces the simulation effort and thus, in turn, reduces the cost of testing that will be incurred otherwise.

Major novelties of this thesis include:

- Exhaustive stimuli simulations are performed for various open and short-circuit intra-cell defect sites of varying resistance magnitudes.
- The need for multi-bit change and multi time frame patterns is exposed.
- An algorithmic approach based on switch-level, Elmore delay model is devised to generate test patterns that detect open and short-circuit intra-cell defects and provide

speed-up.

The rest of the thesis is organized as follows, Chapter 2 presents prior work in the field of fault modeling and testing leading up to this research. Chapter 3 describes the defect characterization based on simulation experiments and their results for both open and short circuit defects. In Chapter 4*, proposed fast test generation algorithms are discussed individually for open and short-circuit defects in detail. Chapter 5*, demonstrate and elaborate comparison between the exhaustive simulation results and the results obtained using proposed guided simulation algorithms. Chapter 6, presents the conclusion to the thesis and possible future directions for research.

* This is a joint work in collaboration with Sujay Pandey.

# CHAPTER 2
# BACKGROUND AND PRIOR WORK

In this chapter, we discuss different types of fault modeling and testing of standard cells and the currently used test methodology: Cell-aware test.

Integrated circuits are implemented by interconnecting combinational (memory-less functional) standard cells such as NAND gate, NOR gate, etc. and sequential (state retaining) standard cells such as latches and flip-flops as its building blocks. These standard cells are available in a package known as Standard cell library. Standard cell library is used in IC design due to its robustness and flexibility, provided by different drive strength cells available for reuse, thus lowering the design time and cost. Usually, each cell in standard cell library consist of a pull-up network of PMOS transistors connected to power supply to implement logic 1 by charging the capacitor at the output (by pulling up the output voltage), and a pull-down network of NMOS transistors connected to ground to implement logic 0 by discharging the capacitor at the output (by pulling down the output voltage). In this research, we focus on characterization and testing of combinational standard cells of different drive strengths for different delays, which is found to be challenging in [9], [10] because of the tremendous amount of time and effort required to perform circuit simulations. In [10], it is demonstrated that it can take up to 650 hours to perform circuit simulations using a large number of processors and large disk space, for characterizing a 14nm technology standard cell library with 800 cells, thus it proposes a faster methodology of approximation and curve fitting. For defect characterization of standard cells, various fault models and testing methodologies are used.

## 2.1 Fault modeling

A fault model is the engineering model of a physical defect that can manifest during the fabrication process of a chip and can cause a non-ideal behavior in the circuit. Faults are caused due to specification mistakes, implementation mistakes, external disturbances or component defects. In this work, we focus on component defects that occur during the manufacturing process. Faults are classified based on duration as:

- Permanent faults: These faults are stable and consistently present in the circuit, for example, a short-circuited connection. These are also referred to as hard or solid faults.
- Transient faults: These faults occur temporarily in the circuit causing a disturbance, due to random effects with sudden spikes like electromagnetic interference, power supply fluctuation, etc.
- Intermittent faults: These faults also occur temporarily in the circuit causing a disturbance but the cause of an intermittent fault is continuous degradation of circuit parameters with use or time.

In this work, we focus on open and short-circuit faults that are permanent faults. There are several fault models shown in Figure 2.1 that are used and discussed in the literature some of these are:

### 2.1.1 Stuck-at fault model

It is the most commonly used fault model. It includes the faults that occur when a gate input, gate output or a wire gets stuck-at-0 (short with the ground) or stuck-at-1 (short with the power supply) permanently, independent of the inputs to the circuit [11], [12].

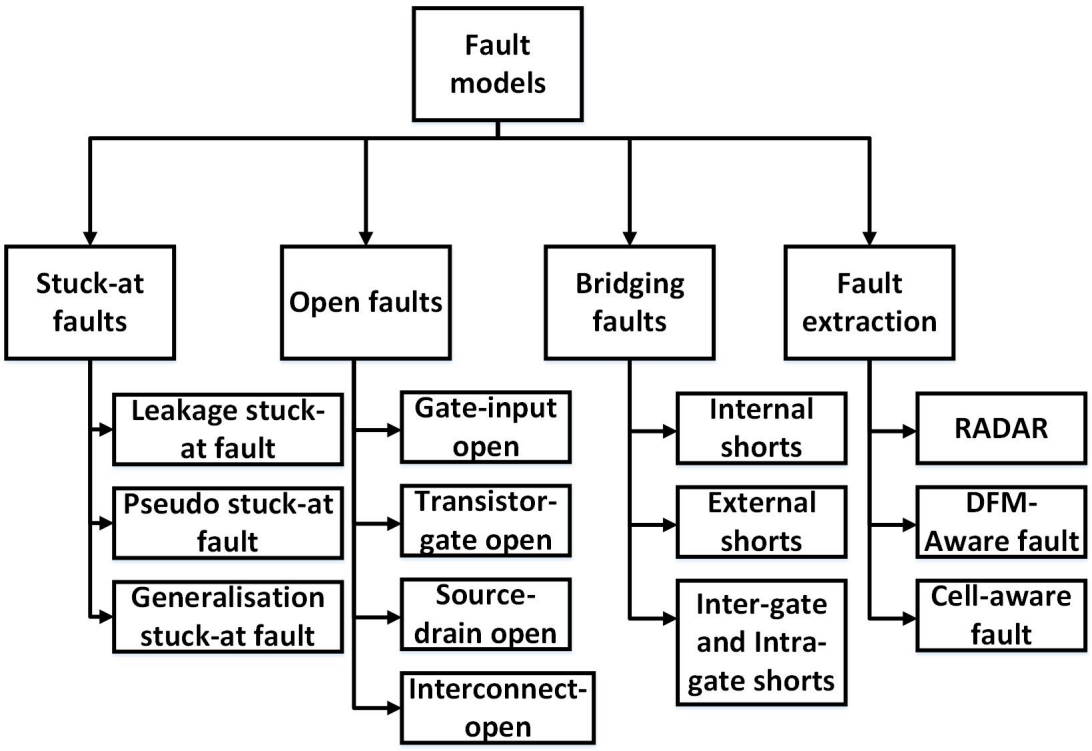


Figure 2.1: Different fault models

### 2.1.2 Open fault model

It includes the faults that occur due to a wire being broken causing one or more floating inputs for the gate that was driven by the broken wire [13]. The circuit behavior depends on the circuit implementation and the inputs applied to it. There are 'n' open faults possible in a circuit with 'n' wires.

### 2.1.3 Bridging fault model

It includes the faults that occur due to wires or signals getting short together causing a non-ideal behavior [11]. Often, it leads to a wired-AND or wired-OR logic implementation depending on the circuit implementation. Since the number of bridging connections possible in a circuit with 'n' wires is n$\times$(n-1), these faults are usually restricted to the bridging faults with higher probability due to adjacency in the physical layout of the circuit design.

### 2.1.4 Transistor fault model

It includes the faults that occur at the transistor level. It can be of stuck-short (where the transistor is short or always-ON) or stuck-open (where the transistor is open or always-OFF) [14]. Stuck-short can produce a short between the power supply and the ground.

### 2.1.5 Delay fault model

It includes the faults in which the signal is able to attain the correct value eventually but it takes a longer time than the ideal or fault-free scenario. Delay fault models can be further categorized as transition delay [12], [14], gate delay and path delay [15].

Testing of these fault models can be challenging due to various reasons. For example, testing path delay can be extremely time complex due to a large number of paths present in the Circuit Under Test (CUT). Therefore, to improve fault coverage other fault modeling schemes such as timing-aware [16], DFM-aware [17], cell-aware [4], N-detect [18], Embedded multi-detect [19] and are researched. Schemes like Gate exhaustive [1] and region-exhaustive [20] are proposed to expose faults within the standard gates used in the circuit design, in the past. They require identification of a set of circuit inputs to be applied to achieve all possible input combinations at cell inputs of the cells used in the circuit, which is computational complex and time expensive. For detection of transistor-level short-circuit defects, commonly used fault models include short between Gate-Drain, Gate-Source and Drain-Source terminals [21] or a Gate Oxide fault model consisting of a short between the gate and the surface of silicon through the dielectric of a transistor [22]. Some test generation and fault modeling schemes are based on switch-level models representing each transistor as a switch and analyzing the charging and the discharging phenomena of the output capacitance and parasitic capacitance present within the cell [23]. Open and short-circuit defects within a cell can cause delay faults as well as erroneous output, thus they need to be modeled and tested.

## 2.2 Testing

For testing a circuit, input patterns are generated, applied and compared with the ideal, fault-free test scenario. According to [24] testing can be classified as:

### 2.2.1 Functional testing

Functional testing is performed to verify the functional correctness of the output of a circuit. It is the first and basic test requirement for a circuit.

### 2.2.2 Dynamic testing

Dynamic testing is performed to verify the response time or delays of a circuit in normal operating conditions. An example of dynamic testing technique is Structural scan testing. For this, serial chains of specialized flip-flops know as scan flip-flops are inserted in the CUT for controllability and observability of testing. These scan chains are fed at low speed, with serially shifting input vectors. The last shift operation is performed at speed, to observe the final output and identify any abnormalities. This technique is popularly used for the transient scan delay test to verify the timing behavior of different paths in a circuit and characterize them.

### 2.2.3 Parametric testing

Parametric testing is performed to verify the electrical properties such as voltage, current, power consumption, etc. of a circuit. An example of parametric testing is IDDQ testing which is performed to test the amount of leakage current in the circuit. The basic principle of IDDQ testing is that at stable state, there is an average amount of leakage current drawn from the power supply but there is no direct connection between the power supply and the ground, so if there is an unusually large current being drawn from the power supply it implicates defect in the transistor device involved. Thus, it is very effective in the detection

of shorts in the circuit [25].

## 2.3 Cell-aware testing

Cell-aware testing (CAT) methodology uses a single defect model to characterizes defects within a cell, by exploiting their behavior under fault-free and faulty condition of the cell using circuit simulation [5]. The set of input stimuli that detect open or short-circuit faults through simulation are applied post-manufacturing of the integrated circuit chip and the circuit behavior is tested. This reduces the test size from all possible input patterns to only cell-aware generated patterns. Such a characterization is effort and time expensive but it is still used for post-manufacturing test pattern generation [26], since it has been proven to have higher fault coverage in comparison to stuck-at and transition-delay fault (TDF) test methodologies [4], [27], [28].

Recently modifications have been made to CAT methodology, focusing on the analysis for small and gross delay effect [29]. Another recent advancement of the timing-aware cell-aware test was made for small delay defects in 14nm finFET technology chips for DPPM reduction [30]. It has been described in [6], that despite CAT methodology some defects are known to escape structural testing and are detected during system-level testing. These faults are majorly due to open and short-circuit defects within the standard cells used in circuits. Therefore, this creates a need for a more sophisticated structural testing methodology.

In general, a test pattern generator generates a set of input vector patterns that are fed to the Circuit Under Test (CUT) post-manufacturing through scan architecture (consisting of scan chains and flip-flops), which was inserted during the design-for-test stage of the chip design flow. The patterns are fed and clocked using a test controller and the output responses are measured and analyzed using a response analyzer.

In this work, we focus on a single fault model of intra-cell (transistor level) open and short-circuit faults. We observe the defect behavior for open and short-circuit defects at

various defect locations with varying magnitudes of defect resistance by performing exhaustive circuit simulations at the cell-level and conclude the necessity of multi-bit change patterns and/or multi-pattern delay test in detecting some defects. Results from simulations are observed and analyzed to identify possible reasons for a functional fault or delay fault. These simulation results are exploited to develop switch-level, Elmore-delay based, fast test generating framework algorithms to generate test vector patterns. These algorithms identify the input vector patterns required to detect a fault in a circuit without the need for exhaustive circuit simulation based on charging and discharging of the cell output capacitor and the parasitic capacitors in a cell netlist. These algorithms also employ concepts of device physics such as charge sharing, contention due to a direct connection between the power supply and ground.

# CHAPTER 3
# SIMULATION STUDIES

In this chapter, we elaborate on the simulation experiments that were performed: the fault model we used for the fault injection in standard cells, types of circuit simulations we performed on these cells and the classification of the results.

We performed experiments on parasitic extracted netlists of standard cells from NCSU 45nm FreePDK NAND-gate [31] standard cell library cells. These cells were provided with a supply voltage of 0.8V for circuit simulations. Fault-free cell netlists were simulated for DC and transition-delay test to obtain the correct (or golden) functional outputs and delays. For generating a faulty cell netlist, resistive open and short-circuit defects of varying magnitudes were injected into the cells at various locations depending on the fault model used. The faulty cell netlists were then simulated for DC and transition-delay tests with the same parameters as the fault-free cell netlist simulation. The resulting output and delays from the faulty cell netlist were compared with golden output and delays. The circuit simulator used for these experiments was HSPICE simulator from Synopsys.

Let a parasitic extracted cell netlist have 'M' MOSFETs, 'C' parasitic capacitances, 'R' parasitic resistances, 'n' inputs and 'm' outputs. Based on these notations, the fault models and the experimental procedure is described below:

## 3.1 Fault Injection

To analyze the partial-open and short-circuit faults that occur post-fabrication at the transistor level, within the cells of a standard cell library, we use the following resistive open and short fault models:

### 3.1.1 Open-circuit fault model

For each cell in the standard cell library, we inject parametric resistive open defects, $R_g$, $R_d$, and $R_s$ at the gate terminal, the drain terminal, and the source terminal, respectively, of each MOSFET present in the parasitic extracted cell netlist, as shown in Figure 3.1. Thus, we have 3M defect sites for M MOSFETs in a cell. Additionally, each of these resistive open defect sites can have 7 possible magnitudes $1k\Omega, 5k\Omega, 10k\Omega, 50k\Omega, 500k\Omega, 500M\Omega, 1G\Omega$ according to our experimental setup. Thus, the fault universe for each cell with open-circuit defects is 21M.

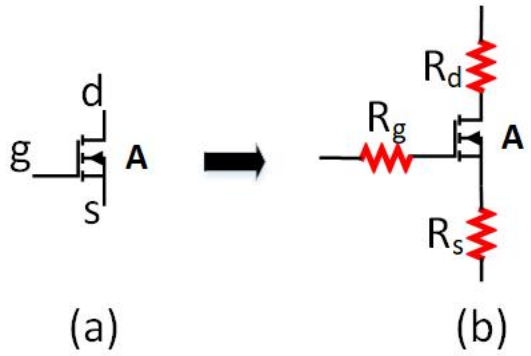


Figure 3.1: Fault injection for open-circuit fault model

### 3.1.2 Short-circuit fault model

There are three types of resistive short fault models that are used in our experiments, as described below:

*DS short-circuit fault model*

For each cell in the standard cell library, we inject a parametric resistive short-circuit defect, $R_{ds}$ between the drain terminal and the source terminal of each MOSFET present in the parasitic extracted cell netlist, as shown in Figure 3.2. Thus, we have M defect sites for M MOSFETs in a cell. Additionally, each of these resistive short-circuit defect sites can have 6 possible magnitudes $\{1k\Omega, 10k\Omega, 20k\Omega, 30k\Omega, 40k\Omega, 50k\Omega\}$ according to our

experimental setup. Thus, the fault universe for each cell with DS short-circuit defects is 6M.

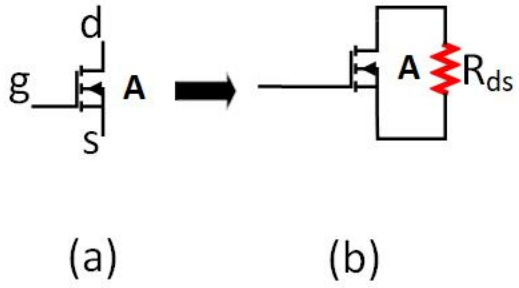


Figure 3.2: Fault injection for DS short-circuit fault model

*GD-GS short-circuit fault model*

For each cell in the standard cell library, we inject two parametric resistive short defect, $R_{gd}$ and $R_{gs}$. $R_{gd}$ is injected between the gate and the drain terminals and $R_{gs}$ is injected between the gate and the source terminals of each MOSFET present in the parasitic extracted cell netlist, as shown in Figure 3.3. Thus, we have 2M fault sites for M MOSFETs in a cell. Additionally, each of these resistive fault sites can have 6 possible values $\{1k\Omega, 10k\Omega, 20k\Omega, 30k\Omega, 40k\Omega, 50k\Omega\}$ according to our experimental setup. Thus, the fault universe for each cell with GD-GS short-circuit faults is 12M.

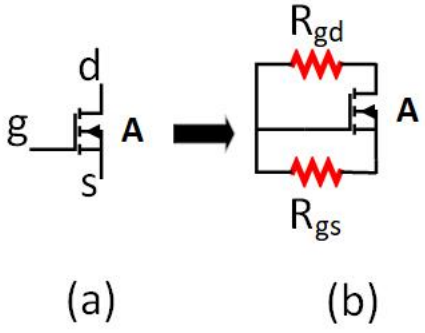


Figure 3.3: Fault injection for GD-GS short-circuit fault model

*Capacitance short-circuit fault model*

For each cell in the standard cell library, we inject a parametric resistive short-circuit defect, $R_{cap}$ between the terminals of each parasitic capacitance present in the parasitic extracted cell netlist, as shown in Figure 3.4. Thus, we have C defect sites for C parasitic capacitances in a cell. Additionally, due to an exploding number of parasitic capacitances present in each cell, each of these resistive short-circuit defect site is limited to have only 4 possible magnitudes $\{1k\Omega, 20k\Omega, 40k\Omega, 50k\Omega\}$ according to our experimental setup. Thus, the fault universe for each cell with capacitance short-circuit defects is 4C.

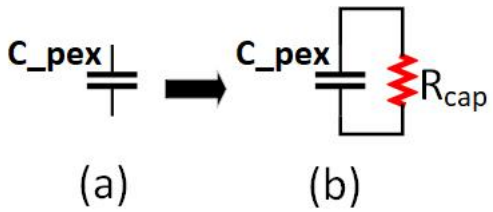


Figure 3.4: Fault injection for Capacitance short-circuit fault model

The magnitude values of these resistive defects for all the fault models were chosen based on initial delay sensitivity observations from preliminary circuit simulation experiments on a few standard cells. Also, we consider the single fault model that assumes the existence of only one of these resistive defects from either of the fault models at a time while performing simulations.

## 3.2 Simulation test bench

The test bench used for circuit simulation in our experimental setup is shown in Figure 3.5. Automatic fault injection of all parametric resistive faults from the fault universe of each fault model for open and short-circuit defects, is performed on the parasitic extracted netlist of standard cells from the standard cell library. Based on the single fault model, only a single fault is active at a time for each simulation. A fault-free inverter is cascaded at the output of each cell to pull up or pull down the intermediate analog values that may

be produced due to the fault in the faulty netlist.

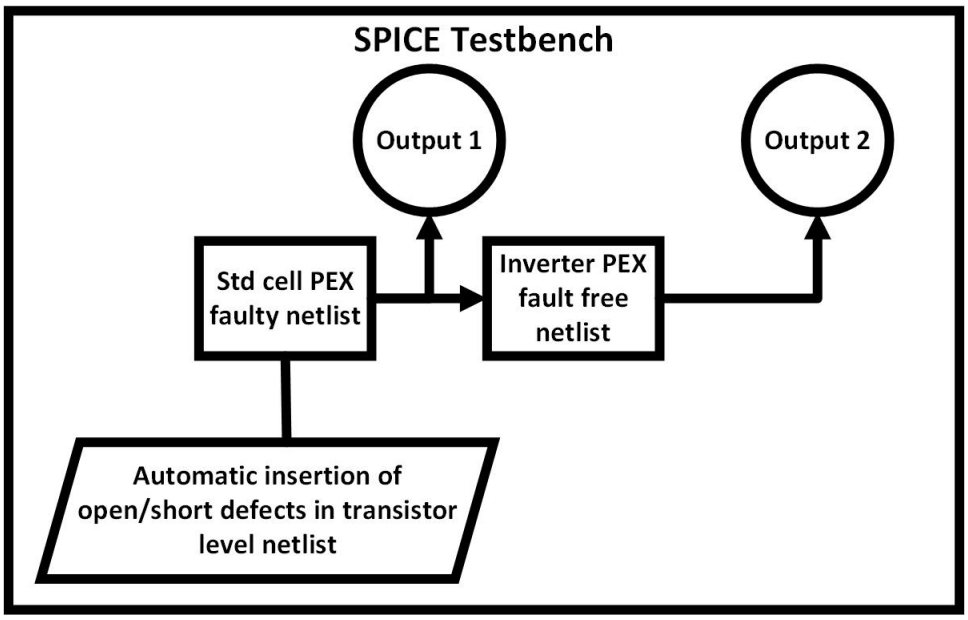


Figure 3.5: Simulation test bench

The circuit simulation is performed in three phases:

- DC simulation and analysis
- Two time frame simulation and analysis
- Three time frame simulation and analysis.

In each phase, simulations for fault-free and faulty parasitic extracted cell netlist are performed, and the cell output values and the output transition delays are analyzed and compared. The details of the three phases of simulation and their analysis are described as follows:

### 3.2.1 DC simulation and analysis

DC simulation is performed to analyze the static voltage characteristics of a standard cell output, in the absence and the presence of a defect. For each cell, these simulations are performed in two phases: fault-free netlist simulation (absence of defect) and faulty netlist

simulation (presence of defect) with an exhaustive set of DC input vectors and then finally, an analysis to performed for fault detection. For a cell with 'n' cell inputs, $2^n$ n-bit DC input vectors are possible for exhaustive simulations.

*Fault-free netlist simulation:*

The fault-free parasitic extracted netlist is subjected to DC simulation for an exhaustive set of $2^n$ n-bit DC input vector stimuli, which are applied to observe the analog voltage values at 'm' cell output signals (output 1) and their corresponding 'm' inverter output signals (output 2) for each DC input vector. The analog value of each signal bit of both the outputs, output 1 and output 2 is normalized to logic 1 if it is above 0.7V and it is normalized to logic 0 if it is below 0.1V, to construct a golden (fault-free) truth table with $2^n$ n-bit input vectors and their corresponding, $2^n$ m-bit cell output vectors and $2^n$ m-bit inverter output vectors.

*Faulty netlist simulation:*

We consider each defect location and defect magnitude depending on the corresponding fault model to generate a faulty parasitic extracted netlist. For each defect location with each defect magnitude, this netlist is subjected to DC simulation for exhaustive set of $2^n$ n-bit DC input vector stimuli, which are applied to observe the analog voltage values at 'm' cell output signals (output 1) and their corresponding 'm' inverter output signals (output 2), for each input vector. The analog value of each signal bit of both the outputs, output 1 and output 2 is normalized to logic 1 if it is above 0.7V and it is normalized to logic 0 if it is below 0.1V, to construct a faulty truth table with $2^n$ n-bit input vectors and their corresponding, $2^n$ m-bit cell output vectors and $2^n$ m-bit inverter output vectors.

*Analysis:*

The m-bit output vectors at output 1 and output 2 of the faulty truth table for each defect (location and magnitude), are compared with the m-bit output vectors at output 1 and output 2 of the golden truth table, for the corresponding input vector. If there is a mismatch in the logic levels between the golden output 1 and the faulty output 1, or the golden output 2 and the faulty output 2 for any input vector, then the fault is said to be detected by the specific input vector stimulus.

### 3.2.2 Two time frame simulation and analysis

To detect the defects from the fault universe, that were not detected in DC analysis, they are subjected to two time frame transient simulation and analysis. Two time frame simulation is performed to analyze transition-delay characteristics of a standard cell output, in the absence and the presence of a defect. For a cell with 'n' cell inputs, $2^n \times (2^n - 1)$ two time frame input vector patterns are possible for application in exhaustive simulation (Note that application of identical vectors in consecutive last two time frames is equivalent to DC simulation, thus we do not consider them in two time frame simulation). These input vector patterns can be denoted as $\{V_1, V_2\}$, where V1 is the first vector and $V_2$ is the second vector. For the transient analysis of the cell, with each input transition from the first input vector $V_1$ to the second input vector $V_2$, three output transition scenarios are possible for each bit in the output vector: the output can either remain static, rise or fall. For example, for a 2-input fault-free NAND gate:

- For an input vector transitioning as $\{V_1, V_2\} = \{01, 10\}$, the output is static at 1
- For an input vector transitioning as $\{V_1, V_2\} = \{11, 00\}$, the output rises from 0 to 1
- For an input vector transitioning as $\{V_1, V_2\} = \{11, 00\}$, the output falls from 1 to 0

Each of these input vector patterns can have $2^n$ possible initial conditions. This results into an exhaustive set of $2^n \times 2^n \times (2^n - 1)$ two time frame input vector patterns, to be

possible for simulation and the input vector pattern are now denoted as $\{V_0, V_1, V_2\}$, where $V_0$ is the initialization vector and $V_1$ and $V_2$ hold the same meaning as before. Two time frame patterns are further classified as:

1. Single Bit Same Initialization (SBSI): These patterns consist of two time frame vector patterns having a single bit change between the last two vectors (which are $V_1$ and $V_2$) with the initialization vector same as the first vector ($V_0 = V_1$), as is prevalent with existing CAT practice. For example, $\{V_0, V_1, V_2\} = \{00, 00, 01\}$ where $V_0 = V_1 = 00$ and $V_2 = 01$ with only a single bit change from $V_1$ to $V_2$.

2. Single Bit Different Initialization (SBDI): These patterns consist of two time frame vector patterns having a single bit change between the last two vectors (which are $V_1$ and $V_2$) with the initialization vector different than the first vector ($V_0 \neq V_1$). For example, $\{V_0, V_1, V_2\} = \{01, 00, 01\}$ where $V_0 = 01$, $V_1 = 00$ and $V_2 = 01$, thus $V_0 \neq V_1$ with only a single bit change from $V_1$ to $V_2$.

3. Multi Bit Same Initialization (MBSI): These patterns consist of two time frame vector patterns having a multi (more than single) bit change between the last two vectors (which are $V_1$ and $V_2$) with the initialization vector same as the first vector ($V_0 = V_1$). For example, $\{V_0, V_1, V_2\} = \{00, 00, 11\}$ where $V_0 = V_1 = 00$ and $V_2 = 11$ with only a multi bit change from $V_1$ to $V_2$.

4. Multi Bit Different Initialization (MBDI): These patterns consist of two time frame vector patterns having a multi (more than single) bit change between the last two vectors (which are $V_1$ and $V_2$) with the initialization vector different than the first vector ($V_0 \neq V_1$). For example, $\{V_0, V_1, V_2\} = \{01, 00, 11\}$ where $V_0 = 01$, $V_1 = 00$ and $V_2 = 11$, thus $V_0 \neq V_1$ with only a multi bit change from $V_1$ to $V_2$.

For each cell, two time frame simulations are performed in two phases: fault-free netlist simulation (absence of defect) and faulty netlist simulation (presence of defect) with an

exhaustive set of two time frame input vector patterns and then finally, an analysis to performed for fault detection.

*Fault-free netlist simulation:*

The fault-free parasitic extracted netlist is subjected to two time frame simulation for an exhaustive set of $2^n \times 2^n \times (2^n - 1)$ two time frame input vector stimuli which are applied to observe and record the transition behavior and delay values for 'm' cell output signals (output 1) and their corresponding 'm' inverter output signals (output 2) for each input vector. For each input vector pattern, if a bit signal of output vector 1 or output vector 2 undergoes a transition (rise or fall), the rise or fall time is measured. The maximum rise time ($T_{r-max(j)}$) and maximum fall time ($T_{f-max(j)}$) among all the measured values in the respective categories is determined for each output bit 'j' from m-bit output vector at output 1. Similarly, maximum rise time ($T_{r-max(k)}$) and maximum fall time ($T_{f-max(k)}$) is determined for each output bit 'k' from m-bit output vector at output 2.

*Faulty netlist simulation:*

We consider each defect location and defect magnitude depending on the corresponding fault model to generate a faulty parasitic extracted netlist. For each defect location with each defect magnitude, this netlist is subjected to two time frame simulation for an exhaustive set of $2^n \times 2^n \times (2^n - 1)$ two time frame input vector stimuli, which are applied to observe and record the transition behavior and delay values for 'm' cell output signals (output 1) and their corresponding 'm' inverter output signals (output 2), for each input vector. For each input vector pattern, if the bits of output vector 1 or output vector 2 undergo a transition (rise or fall), the rise or fall time for both output vectors at output 1 and 2 is measured.

*Analysis:*

The transition behaviour for each bit of the output vectors at output 1 and output 2 from the faulty netlist simulations, for each defect (location and magnitude) is compared with the transition behavior of the corresponding bit in the output vectors at output 1 and output 2 from the fault-free netlist simulation, for the corresponding input vector. If there is a mismatch in the transition behavior for either bit of output 1 or output 2 in the faulty netlist in comparison to the fault-free netlist, then the fault is said to be detected by the specific input vector stimulus. If the transition behaviour for each bit of output 1 and output 2 matches in the fault-free and the faulty netlist simulation, then we check if the transition delay for any bit in output vectors at output 1 or output 2, exceeds the worst-case rise delay (maximum rise delay) or worst-case fall delay (maximum fall delay) corresponding to that particular bit, by more than 20% (we select between worst-case rise delay and worst-case fall delay depending on the transition behavior for the particular bit). If the transition delay does exceed the worst-case rise or fall delay for any bit in either of the output vectors, then the fault is said to be detected by the specific input vector.

### 3.2.3 Three time frame simulation and analysis

To detect the defects from the fault universe, that were not detected in two time frame simulation and analysis, they are subjected to three time frame transient simulation analysis. Three time frame simulation is performed to analyze the increase in fault coverage due to transition-delay characteristics of a standard cell output, in the absence and the presence of a defect, on application multi-time frame (MTF) test pattern. For a cell with 'n' cell inputs, $2^n \times 2^n \times (2^n - 1)$ three time frame input vector patterns are possible for application in exhaustive simulation (Note that application of identical vectors in consecutive last two time frames is equivalent to DC simulation, thus we do not consider them in three time frame simulation). Each of these input vector patterns can have $2^n$ possible initial conditions. This results into an exhaustive set of $2^n \times 2^n \times 2^n \times (2^n - 1)$ three time frame input vector

patterns, possible for simulation and the input vector pattern are denoted as $\{V_0, V_1, V_2, V_3\}$, where $V_0$ is the initialization vector, $V_1$ is the first vector, $V_2$ is the second vector and $V_3$ is the third vector. Similar to two time frame simulation analysis, there are three possible output transition scenarios: the output can either remain static, rise or fall, when input transitions from the second input vector $V_2$ to the third input vector $V_3$.

For each cell, three time frame simulations are performed in two phases: fault-free netlist simulation (absence of defect) and faulty netlist simulation (presence of defect) with an exhaustive set of three time frame input vector patterns and then finally, an analysis to performed for fault detection.

*Fault-free netlist simulation:*

The fault-free parasitic extracted netlist is subjected to three time frame simulation for an exhaustive set of $2^n \times 2^n \times 2^n \times (2^n - 1)$ three time frame input vector stimuli which are applied to observe and record the transition behavior and delay values for 'm' cell output signals (output 1) and their corresponding 'm' inverter output signals (output 2) for each input vector. For each input vector pattern, if a bit signal of output vector 1 or output vector 2 undergoes a transition (rise or fall), the rise or fall time is measured. The maximum rise time ($T_{r-max(j)}$) and maximum fall time ($T_{f-max(j)}$) among all the measured values in the respective categories is determined for each output bit 'j' from m-bit output vector at output 1. Similarly, maximum rise time ($T_{r-max(k)}$) and maximum fall time ($T_{f-max(k)}$) is determined for each output bit 'k' from m-bit output vector at output 2.

*Faulty netlist simulation:*

We consider each defect location and defect magnitude depending on the corresponding fault model to generate a faulty parasitic extracted netlist. For each defect location with each defect magnitude, this netlist is subjected to three time frame simulation for an exhaustive set of $2^n \times 2^n \times 2^n \times (2^n - 1)$ three time frame input vector stimuli which are

applied to observe and record the the transition behavior and delay values for 'm' cell output signals (output 1) and their corresponding 'm' inverter output signals (output 2), for each input vector. For each input vector pattern, if the bits of output vector 1 or output vector 2 undergo a transition (rise or fall), the rise or fall time for both output vectors is measured.

*Analysis:*

The transition behaviour of each bit in the output vector at output 1 and output 2 from the faulty netlist simulation, for each defect (location and magnitude) is compared with the transition behavior of the corresponding bit in the output vectors at output 1 and output 2 from the fault-free netlist simulation, for the corresponding input vector. If there is a mismatch in the transition behavior for either bit of output 1 or output 2 in the faulty netlist in comparison to the fault-free netlist, then the fault is said to be detected by the specific input vector stimulus. If there the transition behaviour for each bit of output 1 and output 2 matches in the fault-free and the faulty netlist simulation, then we check if the transition delay for any bit in output vector of output 1 or 2, exceeds the worst-case rise delay (maximum rise delay) or worst-case fall delay (maximum fall delay) corresponding to that particular bit, by more than 20% (we select between worst-case rise delay and worst-case fall delay depending on the transition behavior for the particular bit). If the transition delay does exceed the worst-case rise or fall delay for any bit in either of the output vectors, then the fault is said to be detected by the specific input vector.

### 3.3 Simulation results

These experiments are conducted to demonstrate the necessity of multi-bit change vector patterns and multi time frame vector patterns in the detection of certain magnitudes of resistive open and short-circuit defects present in a standard cell, thereby increasing the fault coverage. It is observed that with an increase in the complexity of testing from DC

to two time frame to three time frame, more and more faults are detected. Therefore, we classify each fault as uniquely identified by each of these simulation pattern tests: DC pattern test, SBSI pattern test, SBDI pattern test, MBSI pattern test, MBDI pattern test, and MTF test.

In each of these fault models, all defects are first subjected to DC simulation and analysis. If they are not detected DC simulation and analysis, then they are subjected to two time frame simulation and analysis: starting with SBSI patterns, if the faults are not detected by SBSI patterns, they are subjected to SBDI patterns, if not detected by SBDI patterns, they are subjected to MBSI patterns, if not detected by MBSI patterns, they are subjected to MBDI patterns. The defects that not detected by two time frame simulation and analysis are subjected three time frame simulation and analysis (MTF).

### 3.3.1 Open-circuit fault model

Simulation results obtained for the open-circuit fault model are shown in Table 3.1. Column 1 of the table shows the cell names of a few cells from the standard cell library, on which the exhaustive simulation experiments were performed. Column 2 shows the number of MOSFETs present in the parasitic extracted netlist of each cell shown in column 1. As discussed in Section 3.1, for a cell with M MOSFETs, there will be 21M faults in fault universe for open-circuit fault model, as shown in column 3.

The faults detected by DC simulation and analysis for each cell is shown in column 4. It was observed that faults at the gate terminal of a transistor, were not detected by DC analysis and simulation irrespective of fault resistance magnitude. In higher drive strength cells (X2 or X4), since the transistors are connected to parallel, there always exists a fault-free path in parallel to fault-injected path because of the single fault model assumption. This phenomenon can be observed from the table, since the number of detections by DC simulation and analysis are none to less, in case of higher drive strength cells. It is observed that only complete opens or high resistance values, for resistive open-circuit defects, are

Table 3.1: Fault detection for open-circuit faults

| Cell | M | Faults | DC | SBSI | SBDI | MBSI | MBDI | MTF |
|---|---|---|---|---|---|---|---|---|
| **OR2_X1** | 6 | 126 | 16 | 85 | 4 | 4 | 4 | 7 |
| **OR2_X2** | 8 | 168 | 8 | 127 | 6 | 8 | 8 | 3 |
| **OR2_X4** | 16 | 336 | 0 | 208 | 28 | 49 | 13 | 9 |
| **OR3_X2** | 10 | 210 | 12 | 162 | 21 | 13 | 1 | 1 |
| **NOR3_X2** | 12 | 252 | 0 | 179 | 1 | 0 | 2 | 2 |
| **AND3_X1** | 8 | 168 | 20 | 125 | 6 | 3 | 5 | 4 |
| **AND3_X2** | 10 | 210 | 18 | 162 | 14 | 9 | 2 | 1 |
| **AND3_X4** | 20 | 420 | 0 | 353 | 19 | 33 | 6 | 7 |
| **AND2_X4** | 16 | 336 | 0 | 260 | 25 | 3 | 3 | 5 |
| **NAND2_X4** | 16 | 336 | 0 | 122 | 11 | 15 | 20 | 4 |
| **AOI21_X1** | 6 | 126 | 2 | 98 | 0 | 2 | 2 | 1 |
| **XNOR2_X2** | 16 | 336 | 16 | 177 | 1 | 3 | 2 | 3 |
| **HA_X1** | 16 | 336 | 60 | 180 | 8 | 10 | 5 | 27 |

detected by DC simulation and analysis.

Column 5 shows the faults that escaped DC testing but were detected by SBSI pattern during two time frame simulation and analysis. It can be observed that a large majority of faults are detected using SBSI patterns. Column 6 shows the faults that escaped DC and SBSI testing but were detected using SBDI pattern during two time frame simulation and analysis. Similarly, column 7 shows the faults that escaped DC, SBSI, and SBDI testing but were detected using MBSI pattern during two time frame simulation and analysis and column 8 shows the faults that escaped DC, SBSI, SBDI, and MBSI testing but were detected using MBDI pattern during two time frame simulation and analysis. Column 9 shows the faults that escaped DC and all four categories of two time frame simulation and analysis (SBSI, SBDI, MBSI, and MBDI) but were detected by three time frame simulation and analysis (MTF). We observed that higher magnitudes of open-defect resistance were detected using DC patterns and two time frame patterns whereas lower magnitudes required the application of three time frame patterns for their detection.

We also identified that a few gate defects were detected using three time frame simulation and analysis. For example, resistive open-circuit defect R_f in OR2_X4 gate of the

standard cell library is detected by $\{V_0, V_1, V_2, V_3\}$ = {11, 11, 11, 00}, is shown in Figure 3.6.

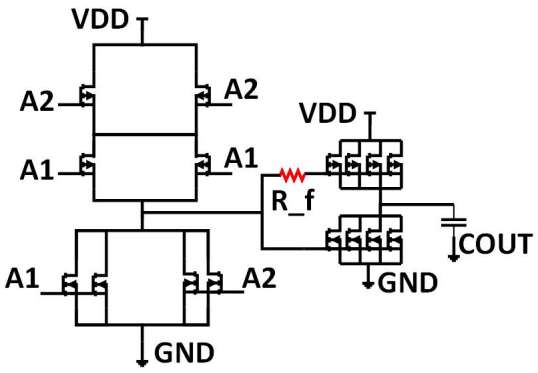


Figure 3.6: OR2_X4 gate fault

A possible hypothesis for this effect is that, due to this resistive open-circuit defect at the gate of a transistor, multiples patterns are required to charge the gate capacitance of the faulty transistor, and thus it is only detected by a three time frame pattern.

From Table 3.1, we can observe a significant number of faults that are detected by multi-bit change or multi time frame vectors patterns, that would be missed otherwise, the quantifying the need for these patterns for detecting additional faults and increasing the fault coverage.

### 3.3.2 Short-circuit fault model

The simulation results and observations for each of the short-circuit fault models, are discussed as follows:

*DS short-circuit fault model*

The simulation results obtained for DS short-circuit fault model are shown in Table 3.2. Column 1 of the table shows the cell names of a few cells from the standard cell library, on which the exhaustive simulation experiments were performed. Column 2 shows the number of MOSFETs present in the parasitic extracted netlist of each cell shown in column 1. As

discussed in Section 3.1, for a cell with M MOSFETs, there will be 6M faults in fault universe for DS short-circuit fault model, as shown in column 3. The faults detected by DC simulation and analysis for each cell is shown in column 4. It is observed that only complete shorts or low resistance values for resistive short-circuit defects, are detected by DC simulation and analysis.

Table 3.2: Fault detection for DS short-circuit faults

| Cell | M | Faults | DC | SBSI | SBDI | MBSI | MBDI | MTF |
|---|---|---|---|---|---|---|---|---|
| **AND2_X1** | 6 | 36 | 8 | 13 | 0 | 0 | 1 | 0 |
| **AOI21_X1** | 6 | 36 | 13 | 11 | 0 | 1 | 0 | 0 |
| **NAND2_X4** | 16 | 96 | 16 | 0 | 0 | 0 | 0 | 0 |
| **OR2_X2** | 8 | 48 | 10 | 12 | 0 | 2 | 0 | 2 |
| **AND3_X1** | 8 | 48 | 13 | 24 | 1 | 0 | 0 | 0 |
| **AND3_X2** | 10 | 60 | 10 | 19 | 7 | 0 | 15 | 0 |
| **AOI21_X2** | 12 | 72 | 16 | 16 | 0 | 6 | 0 | 1 |
| **NAND3_X4** | 24 | 144 | 12 | 12 | 0 | 0 | 12 | 0 |
| **OR3_X1** | 8 | 48 | 16 | 10 | 0 | 4 | 0 | 0 |
| **OR3_X2** | 10 | 60 | 12 | 16 | 2 | 0 | 12 | 3 |

Column 5 shows the faults that escaped DC testing but were detected by SBSI pattern two time frame simulation and analysis. Column 6 shows the faults that escaped DC and SBSI testing but were detected SBDI pattern two time frame simulation and analysis. Similarly, column 7 shows the faults that escaped DC, SBSI, and SBDI testing but were detected using MBSI pattern two time frame simulation and analysis and column 8 shows the faults that escaped DC, SBSI, SBDI, and MBSI testing but were detected using MBDI pattern two time frame simulation and analysis. Column 9 shows the faults that escaped DC and all four categories of two time frame simulation and analysis (SBSI, SBDI, MBSI, and MBDI) but were detected by three time frame simulation and analysis (MTF).

It was observed that lower magnitudes of resistance were detected using low complexity test (DC simulation and analysis), as the magnitude of short resistive defect is increased, it requires a high complexity test to be detected. From the table, we can observe a significant number of faults that are detected by multi-bit change or multi time frame vectors patterns,

that would be missed otherwise, the quantifying the need for these patterns for detecting additional faults and increasing the fault coverage.

*GD-GS short-circuit fault model*

The simulation results obtained for GD-GS short-circuit fault model are shown in Table 3.3. Column 1 of the table shows the cell names of a few cells from the standard cell library, on which the exhaustive simulation experiments were performed. Column 2 shows the number of MOSFETs present in the parasitic extracted netlist of each cell shown in column 1. As discussed in Section 3.1, for a cell with M MOSFETs, there will be 12M faults in fault universe for GD-GS short-circuit fault model, as shown in column 3. The faults detected by DC simulation and analysis for each cell is shown in column 4. It is observed that only complete shorts or low resistance values for resistive short-circuit defects, are detected by DC simulation and analysis.

Table 3.3: Fault detection for GD-GS short-circuit faults

| Cell | M | Faults | DC | SBSI | SBDI | MBSI | MBDI | MTF |
|---|---|---|---|---|---|---|---|---|
| **AND2_X1** | 6 | 72 | 13 | 29 | 2 | 0 | 2 | 0 |
| **AOI21_X1** | 6 | 72 | 32 | 17 | 0 | 1 | 0 | 1 |
| **NAND2_X4** | 16 | 192 | 20 | 4 | 0 | 0 | 0 | 0 |
| **OR2_X2** | 8 | 96 | 22 | 28 | 0 | 2 | 5 | 5 |
| **AND3_X1** | 8 | 96 | 21 | 40 | 6 | 0 | 4 | 1 |
| **AND3_X2** | 10 | 120 | 20 | 38 | 18 | 4 | 6 | 5 |
| **AOI21_X2** | 12 | 144 | 36 | 24 | 0 | 16 | 5 | 1 |
| **NAND3_X4** | 24 | 288 | 20 | 16 | 0 | 0 | 11 | 5 |
| **OR3_X1** | 8 | 96 | 37 | 32 | 0 | 1 | 1 | 2 |
| **OR3_X2** | 10 | 120 | 28 | 50 | 0 | 1 | 2 | 22 |

Column 5 shows the faults that escaped DC testing but were detected by SBSI pattern two time frame simulation and analysis. Column 6 shows the faults that escaped DC and SBSI testing but were detected SBDI pattern two time frame simulation and analysis. Similarly, column 7 shows the faults that escaped DC, SBSI, and SBDI testing but were detected using MBSI pattern two time frame simulation and analysis and column 8 shows

the faults that escaped DC, SBSI, SBDI, and MBSI testing but were detected using MBDI pattern two time frame simulation and analysis. Column 9 shows the faults that escaped DC and all four categories of two time frame simulation and analysis (SBSI, SBDI, MBSI, and MBDI) but were detected by three time frame simulation and analysis (MTF).

It was observed that lower magnitudes of resistance were detected using low complexity test (DC simulation and analysis), as the magnitude of short resistive defect is increased, it requires a high complexity test to be detected. From the table, we can observe a significant number of faults that are detected by multi-bit change or multi time frame vectors patterns, that would be missed otherwise, the quantifying the need for these patterns for detecting additional faults and increasing the fault coverage.

*Capacitance short-circuit fault model*

Through the exhaustive simulation for parasitic capacitance (Cap) short-circuit fault model, it is observed that a large majority of detected faults exist between the cell output or an internal node in the cell and the ground. These faults are detectable by DC, SBSI or SBDI pattern test, and are equivalent to output stuck-at-0 or node stuck-at-0 fault. A large majority of cells from the standard cell library show this behavior, except for a few cells shown in Table 3.4, in which defects at other locations are detectable but they are only detectable by MBSI, MBDI or MTF.

Table 3.4: Fault detection for parasitic capacitance (Cap) short-circuit faults

| Cell | C | Faults | DC | SBSI | SBDI | MBSI | MBDI | MTF |
|---|---|---|---|---|---|---|---|---|
| **AND2_X1** | 38 | 152 | 12 | 11 | 20 | 0 | 1 | 0 |
| **OR2_X2** | 43 | 172 | 14 | 14 | 0 | 0 | 13 | 13 |
| **AND3_X1** | 53 | 212 | 25 | 55 | 0 | 1 | 0 | 1 |
| **AND3_X2** | 52 | 208 | 17 | 17 | 0 | 1 | 9 | 131 |
| **OR3_X2** | 53 | 212 | 18 | 52 | 0 | 2 | 3 | 9 |

In Column 1 of the Table 3.4, the cell names of a few cells that detect some faults using MBSI, MBDI or MTF test patterns. Column 2 shows the number of parasitic capacitors

present in the parasitic extracted netlist of each cell shown in column 1. As discussed in Section 3.1, for a cell with C capacitors, there will be 4C faults in fault universe for Capacitance short-circuit fault model, as shown in column 3. The faults detected by DC simulation and analysis for each cell is shown in column 4. Column 5 shows the faults that escaped DC testing but were detected using SBSI pattern two time frame simulation and analysis. It can be observed that a large majority of detectable faults are detected using DC or SBSI test.

Column 6 shows the faults that escaped DC and SBSI testing but were detected by SBDI pattern two time frame simulation and analysis. Similarly, column 7 shows the faults that escaped DC, SBSI, and SBDI testing but were detected using MBSI pattern two time frame simulation and analysis and column 8 shows the faults that escaped DC, SBSI, SBDI, and MBSI testing but were detected using MBDI pattern two time frame simulation and analysis. Column 9 shows the faults that escaped DC and all four categories of two time frame simulation and analysis (SBSI, SBDI, MBSI, and MBDI) but were detected three time frame simulation and analysis (MTF).

These experiments were conducted in parallel, using a fully automated infrastructure. The time to run these exhaustive simulation increase with the inputs and size of the cell along with the increase in fault universe. For a standard cell library with a large number of cells, exhaustive simulation can be tremendously time and cost expensive.

# CHAPTER 4

# FAST TEST GENERATION ALGORITHMS

As we discussed in Chapter 3 that the time and cost for performing exhaustive analog circuit simulations can exponentially grow with the number of cell inputs and the size of a cell. Additionally, the consideration of a range of resistive defect locations with different defect magnitudes, can further increase the number of circuit simulations to be performed. To circumvent this problem, we propose systematic test generation approaches depending on the different fault models we used. These test generation algorithms analyze the switch-level transistor model for cell netlist using the Elmore delay model to identify the input test stimuli for detecting defects within the cell. For each of these algorithms, we consider a graphical data structure G. An example of graphical representation G, extracted from circuit netlist describing a CMOS cell with complementary pull-up and pull-down network, for 2-input CMOS NAND gate implementation is shown in Figure 4.1. The edges of these graphs denote a transistor and the nodes of the graph represent cell nodes.

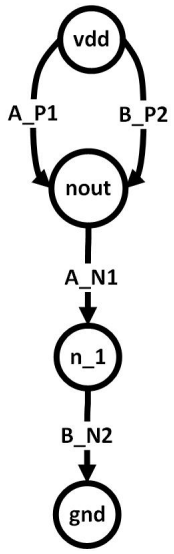


Figure 4.1: Graphical representation for 2-input CMOS NAND gate

Using this graph G, the cell netlist NL and the fault universe FS (depending on the fault model), these algorithms generate input vector patterns to detect defects from the fault universe.

## 4.1 Algorithm for Open-circuit faults

For the open-circuit fault model, we initially consider resistive open defects R1-R8, only at the source terminal and the drain terminal of each transistor in a cell, as shown in Figure 4.2. This maps to 2 defects corresponding to each edge of the graph, defining the fault universe FS under consideration, with only a single defect active at a time.

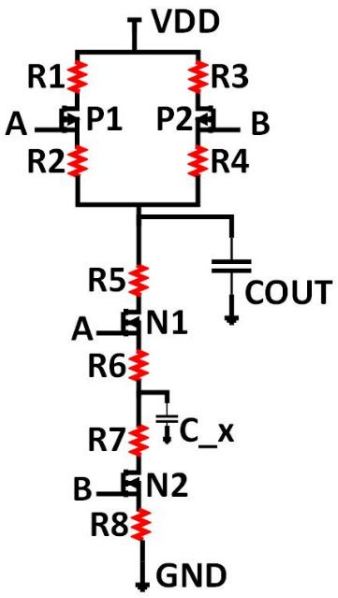


Figure 4.2: Open-circuit fault injection in 2-input CMOS NAND gate

Each cell input vector $V_{in}$ is categorized as either a charging vector (if it charges the cell output capacitance COUT to VDD) or a discharging vector (if it discharges the cell output capacitance COUT to Gnd) in the fault-free scenario. Also, an Elmore delay $ED_{in}$ consistent with the charging or the discharging time of COUT is associated with each input vector $V_{in}$. Along with COUT, there can be other parasitic capacitances which get charged or discharged by the application of $V_{in}$ due to charge sharing, these capacitances

are grouped along with COUT in set $C_{in}$. For example, in a 2-input CMOS NAND gate implementation shown in Figure 4.3, for $V_{in}$ = AB = 10, the Elmore delay $ED_{in}$ associated with charging of COUT, considers the sub-network including R3, R4, COUT, R5, R6, C_x and $C_{in}$ = {COUT, C_x}.

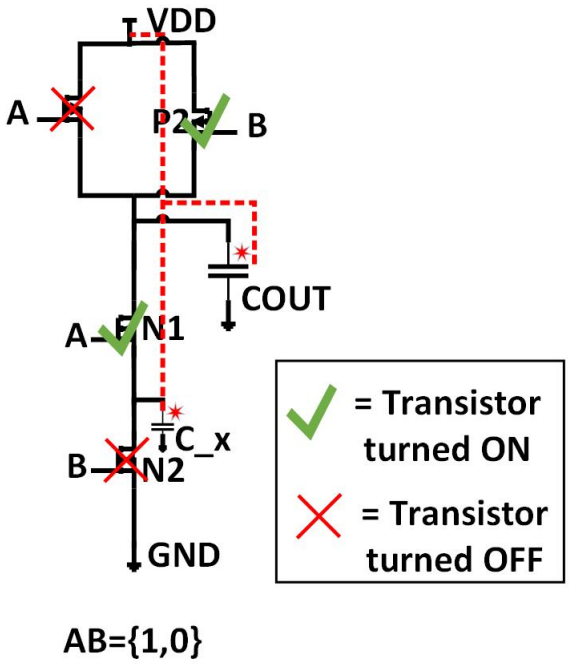


Figure 4.3: $C_{in}$ for $V_{in}$ for 2-input CMOS NAND gate

A basic flow diagram for the test pattern generation, is shown in Figure 4.4. The inputs to the algorithm are fault universe FS and netlist NL using which the algorithm determines an input vector pair $\{V_1, V_2\}$ to maximizes the charging or discharging time of COUT by application of $V_2$ after $V_1$ depending on the $C_1$ and $C_2$ associated with them. We then generate a three pattern test based on maximum transistor switching activity. Each of these steps is elaborated in detail using the pseudo-code shown Algorithm 1.

According to Algorithm 1, for each defect from the fault universe FS, we find the input test vector to switch OFF the transistors in parallel to faulty transistor, if any, using the graph G. Next, for the worst-case charging or discharging Elmore delay, if the defect is in pull-down network, the input vector $V_2$ must discharge the maximum possible number

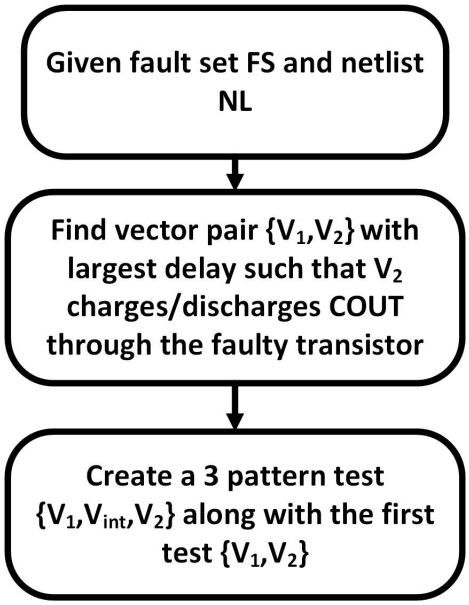


Figure 4.4: Proposed test generation flow for open-circuit faults

of internal node capacitances and the cell output capacitance ($C_2$) along the faulty path in G, and the input vector $V_1$ must charge the maximum possible number of internal node capacitances from the set $C_2$ along with the cell output ($C_1 \cap C_2$ is maximum). Similarly, if the defect is in pull-up network, the input vector $V_2$ must charge the maximum possible number of internal node capacitances and the cell output capacitance ($C_2$) along the faulty path in G, and the input vector $V_1$ must discharge the maximum possible number of internal node capacitances from the set $C_2$ along with the cell output ($C_1 \cap C_2$ is maximum). If several input vectors are applicable, then the first vector from the list of vectors is chosen.

To find a test for the resistive open defect R3 in the example shown in Figure 4.2, we switch OFF transistor P1, which is parallel to faulty transistor P2 by input A = 1. Since the defect is in the pull-up network, we determine from a list of charging vectors generated by the algorithm, an input vector $V_2$, which charges the maximum number of internal node capacitance and COUT, through the faulty transistor (P2). Therefore, for worst-case Elmore delay for charging COUT to VDD, $V_2$ = AB = 10 which charges $C_2$ = {C_x, COUT}, as shown in Figure 4.5(b). Next, we need to find an input vector $V_1$ from a list of discharging vectors generated by the algorithm, such that it discharges the maximum

**stimulus_generation**(NL, FS)
Create graph G for netlist NL;
**for** *(all defects in FS)* **do**
  Inject defect into graph G;
  Set necessary inputs to disable all edges of G parallel to the faulty edge;
  **if** *defect in pull_down network* **then**
    Find discharging vector $V_2$ such that:;
      $ED_2$ is maximum;
    Find charging vector $V_1$ such that:;
      $C_1 \cap C_2$ is maximum;
    $V_{int}$ = Vector that turns on as many transistors in the pull-up chain,
     connected to COUT, as possible;
  **else**
    Find charging vector $V_2$ such that:;
      $ED_2$ is maximum;
    Find discharging vector $V_1$ such that:;
      $C_1 \cap C_2$ is maximum;
    $V_{int}$ = Vector that turns on as many transistors in the pull-down chain,
     connected to COUT, as possible;
  **end**
  two_patterns.append$\{V_1, V_2\}$;
  three_patterns.append$\{V_1, V_{int}, V_2\}$;
**end**

**Algorithm 1:** Test generation algorithm for open-circuit fault model

number of capacitors from $C_2$. Therefore, $V_1$ = AB = 11 which discharges $C_1 = \{$C_x, COUT$\}$, as shown in Figure 4.5(a). This gives the two time frame pattern for detecting the R3 fault as $\{V_1, V_2\} = \{11, 10\}$.

It was observed in [32] from simulation, that although the two vector pattern $\{V_1, V_2\}$ is associated with maximum charging or discharging Elmore delay, it is unable to excite the worst-case delay, for which an intermediate input vector $V_{int}$ is applied after $V_1$ but before $V_2$ thus constituting a three vector pattern $\{V_1, V_{int}, V_2\}$. According to Algorithm 1, $V_{int}$ is selected to provide maximum activity from $V_{int}$ to $V_2$, i.e. if the defect exists in the pull-down network, then $V_{int}$ must switch ON maximum possible number of transistors in the pull-up network and if the defect exists in the pull-up network, then $V_{int}$ must switch OFF maximum possible number of transistors in the pull-down network. This three vector pattern aggravates the output transition delay when input transitions from $V_{int}$ to $V_2$, by

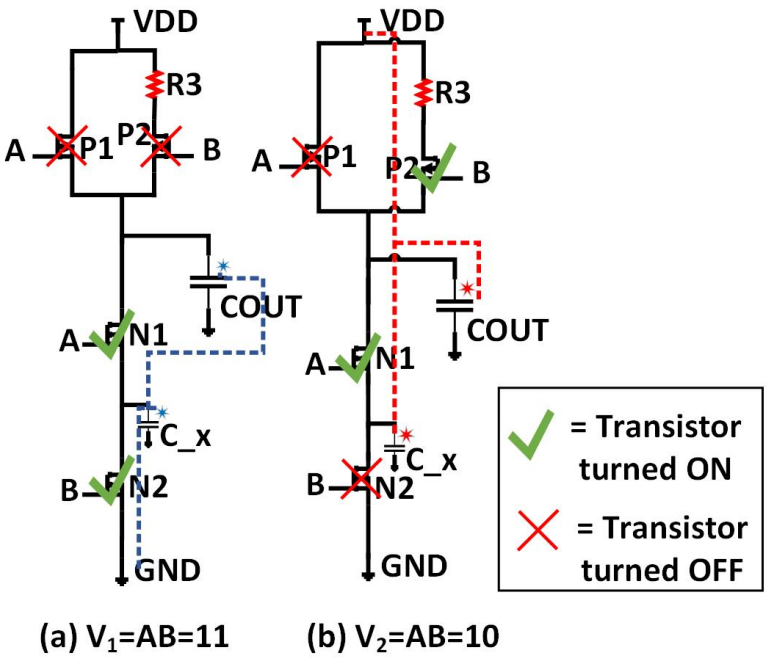


Figure 4.5: Test stimulus for R3 open-circuit fault in 2-input CMOS NAND gate

creating maximum contention between the pull-down and pull-up networks. It is observed that the intermediate pattern is dominant when $V_2$ is either 000..0 or 111..1 for AND, OR, NAND and NOR gates.

***Algorithm results***: Every fault from the fault universe for open-circuit faults can be detected by multiple patterns, during exhaustive analog simulations. If at least a single pattern from the pattern set detecting a fault during exhaustive simulation exists, among patterns generated by the Algorithm 1 for detecting all the faults from the fault universe, then the fault is said to be detected by the algorithm. This analysis was performed to observe the coverage of faults attained by simulating only patterns predicted by the Algorithm 1. The results obtained from this analysis are shown in Table 4.1.

In Table 4.1, column 1 shows the cell names of a few cells from the standard cell library for which algorithm coverage analysis was performed. Column 2 shows the number of MOSFETs present in the parasitic extracted netlist of each cell (shown in column 1). As discussed in Section 3.1, for a cell with M MOSFETs, there will be 21M faults in fault

Table 4.1: Fault detection for open-circuit fault using Algorithm 1

| Cell name | M | Faults | DC | SBSI | SBDI | MBSI | MBDI | MTF |
|---|---|---|---|---|---|---|---|---|
| **OR2_X1** | 6 | 126 | 16 | 82 | 2 | 4 | 4 | 7 |
| **OR2_X2** | 8 | 168 | 8 | 117 | 2 | 8 | 8 | 2 |
| **OR2_X4** | 16 | 336 | 0 | 198 | 11 | 49 | 13 | 8 |
| **OR3_X2** | 10 | 210 | 12 | 143 | 10 | 7 | 1 | 1 |
| **NOR3_X2** | 12 | 252 | 0 | 179 | 1 | 0 | 2 | 2 |
| **AND3_X1** | 8 | 168 | 20 | 120 | 4 | 1 | 4 | 2 |
| **AND3_X2** | 10 | 210 | 18 | 159 | 12 | 5 | 2 | 0 |
| **AND3_X4** | 20 | 420 | 0 | 344 | 19 | 31 | 3 | 4 |
| **AND2_X4** | 16 | 336 | 0 | 256 | 23 | 3 | 3 | 5 |
| **NAND2_X4** | 16 | 336 | 0 | 105 | 5 | 15 | 20 | 4 |

universe for the open-circuit fault model, as shown in column 3. The faults detected by DC patterns through exhaustive simulations, are shown in column 4 (since DC simulations are not expensive, the algorithm was not used for DC test pattern generation). Column 5 shows the faults that escaped DC testing but were detected by algorithm-generated SBSI pattern. Column 6 shows the faults that escaped DC and SBSI testing but were detected by algorithm-generated SBDI pattern. Similarly, column 7 shows the faults that escaped DC, SBSI, and SBDI testing but were detected by algorithm-generated MBSI pattern, and column 8 shows the faults that escaped DC, SBSI, SBDI, and MBSI testing but were detected by algorithm-generated MBDI pattern. Column 9 shows the faults that escaped DC and all four categories of two time frame pattern (SBSI, SBDI, MBSI, and MBDI) but were detected by algorithm-generated three time frame pattern (MTF).

### 4.1.1 Extension of Algorithm 1

Many cells are designed using mirror logic or design techniques other than complementary CMOS logic design. In such a case, Algorithm 1 may not be sufficient due to the absence of a test vector pair that charges and discharges the same internal node capacitances. To generate a multi-pattern test for circuits with mirror logic implementation, we propose a methodology. The flow diagram of the proposed test generation approach, is shown in

Figure 4.6.

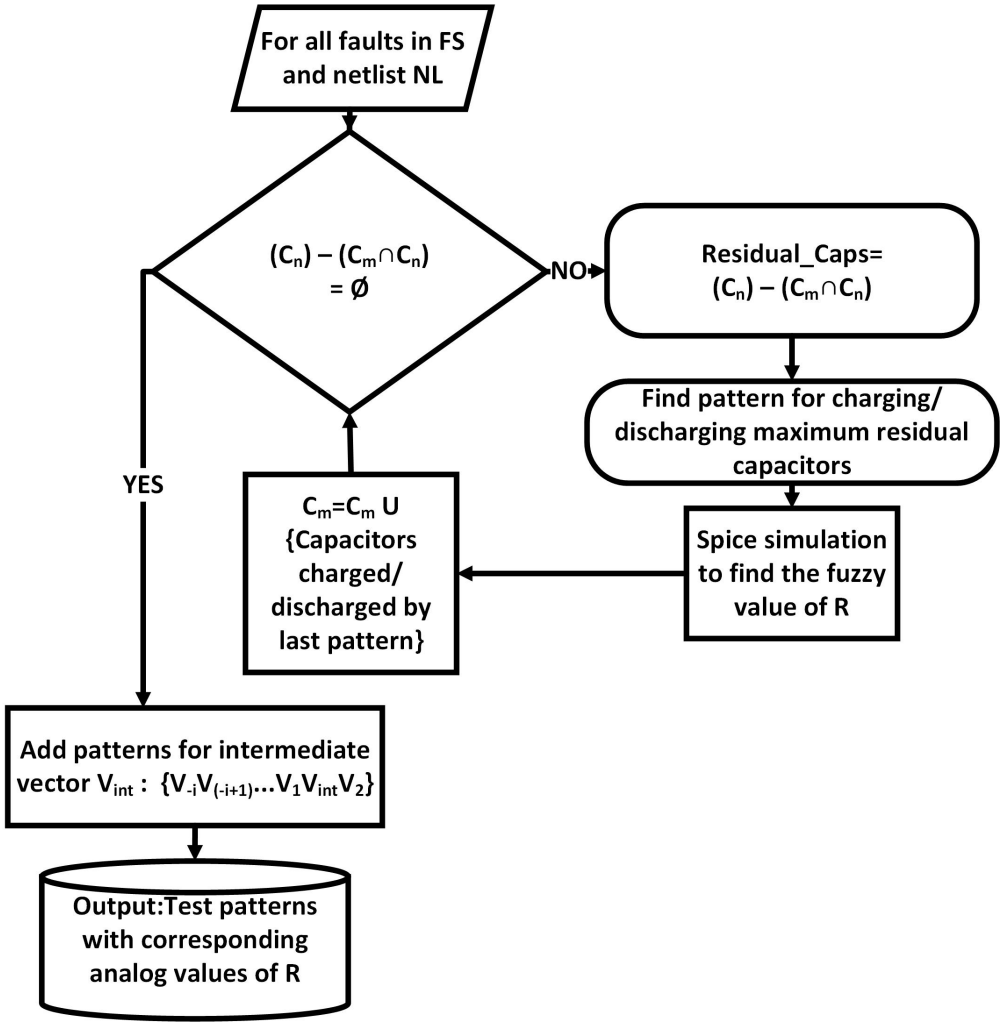


Figure 4.6: Extension of proposed test generation flow for open-circuit faults

Consider the example of a 2-input mirror XOR gate circuit with resistive open defects R1-R16 injected at the source terminal and the drain terminal of each MOSFET present in the cell, as shown in Figure 4.7(a). The graphical representation of the circuit, is shown in Figure 4.7(b). To find the test patterns that detect defect R15, we first identify the last input vector $V_3$, that activates the discharging path (since the defect is in pull-down network) through R15, among all discharging paths for the circuit, with the worst-case Elmore delay $ED_3$ by discharging capacitances in set $C_3$. For this example, $V_3$ = AB = 11 and $C_3$ = {C2, COUT, C3}, as shown in Figure 4.8(c). Next, we find a charging vector $V_2$, which charges

maximum possible capacitances from $C_3$ (such that $C_2 \cap C_3$ is maximum). Therefore for this example, $V_2$ = AB = 01 and $C_2$ = {C2, COUT, C4}, as shown in Figure 4.8(b). The two time frame pattern $\{V_2, V_3\}$ = {01, 11} is not sufficient for detection of defect since there exist residual capacitance: $C_{res\_caps}$ = $C_3$ - ($C_1 \cap C_2$) = {C3}. The presence of residual capacitance implies the total capacitance discharged by the last vector is less than what it possibly can be, thus below a certain analog value for defect resistance R15, the critical delay will not be violated and the defect will not be detected, by the two time frame vector pattern $\{V_2, V_3\}$.

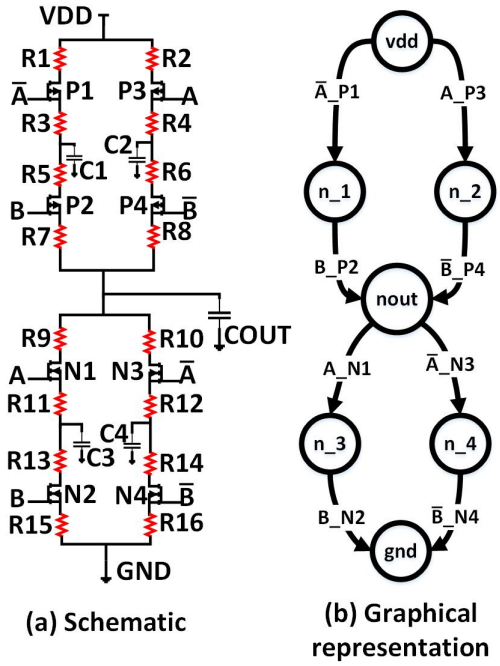


Figure 4.7: Open-circuit fault-injected 2-input mirror XOR gate

We performed, a limited number of circuit simulations with a range of values of R15 around an approximate value, determined by charging and discharging Elmore delay equations for the input vector pattern $\{V_2, V_3\}$, to find the analog value of R15 that escapes the two pattern test $\{V_2, V_3\}$. This value of R15 is found to be $2.16K\Omega$ for the circuit schematic shown in Figure 4.7. For more accurate estimates of open resistance values, a post-layout parasitic extracted netlist should be used. For open defect resistance value using a parasitic

extracted netlist, we observed a small degree of non-monotonic behavior in the relation between the transition delay and open resistance value, which needs to be accounted for in the determination of the critical value of open resistive faults.

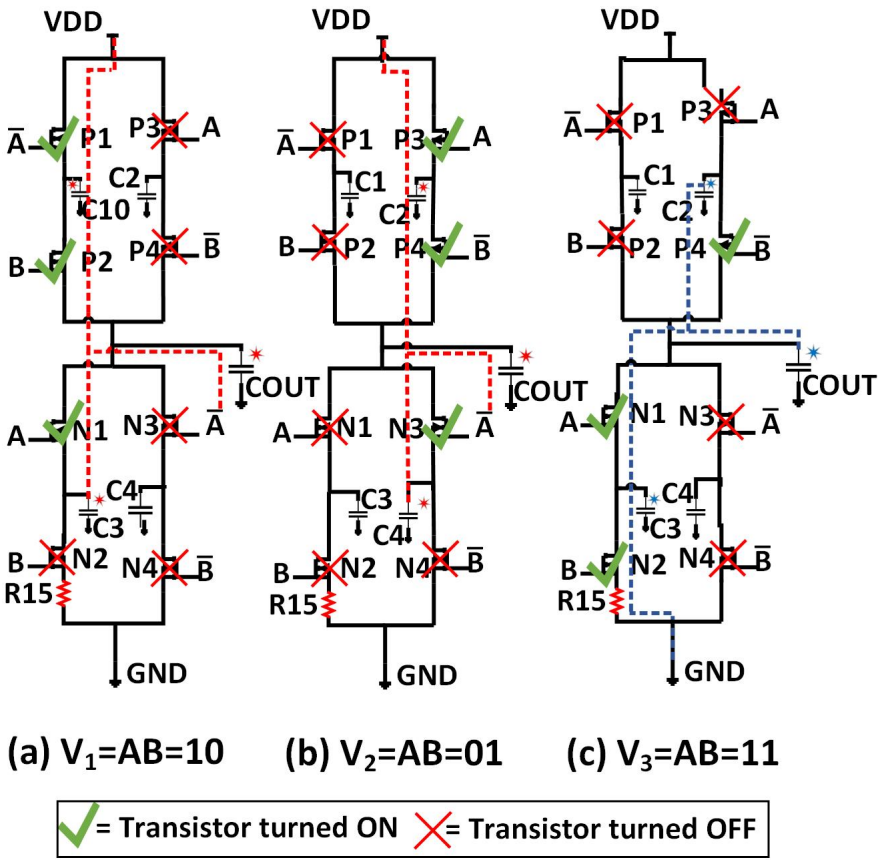


Figure 4.8: Three pattern test for 2-input mirror XOR gate

Since the two time frame test is not sufficient, we proceed to find an input vector $V_1$, that would charge all or maximum possible residual capacitors (as $V_3$ will discharge them). For this example, $V_1$ = AB = 10 since $C_{res\_caps}$ = {C3}, as shown in Figure 4.8(a). The residual capacitors charged by $V_1$ are removed from the set, then we check if any residual capacitors are left to be charged, if not then this approach would be repeated till no residual capacitor is left to be charged. This is done to ensure that all residual capacitors can be charged by a series of input vectors and then discharged by the last input vector. In this example three time frame vector pattern $\{V_1, V_2, V_3\}$ = {10, 01, 11} is sufficient to ensure

this condition. Based on this approach, we can justify that defects at different locations with different defect magnitudes, can need different time frame vectors patterns for their detection.

## 4.2 Algorithm for Short-circuit faults

For short-circuit fault models, we develop two different test pattern generation algorithms, for DS short-circuit defects and GD-GS short-circuit defects based on their delay behavior.

### 4.2.1 Algorithm for DS short-circuit faults

For DS short-circuit fault model, we consider resistive short defects R1-R4, between the drain terminal and the source terminal of each transistor in a cell, as shown in Figure 4.9. This maps to a single defect corresponding to each edge of the graph, defining the fault universe FS under consideration, with only a single defect active at a time.

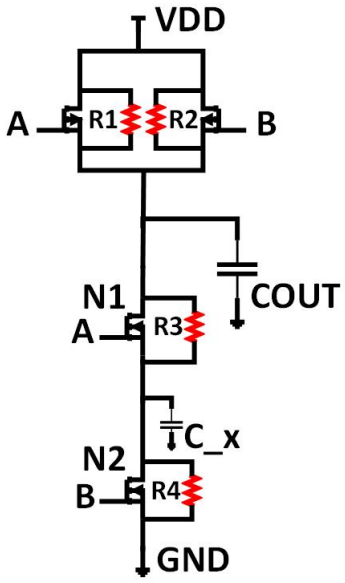


Figure 4.9: DS short-circuit fault injection in 2-input CMOS NAND gate

Each input vector $V_{in}$, is categorized as either a charging vector (if it charges the logic-level output capacitance, COUT to VDD) or a discharging vector (if it discharges the logic-

level output capacitance, COUT to Gnd) under the fault-free scenario. Also, an Elmore delay $ED_{in}$ consistent with the charging or the discharging time of COUT is associated with each input vector $V_{in}$.

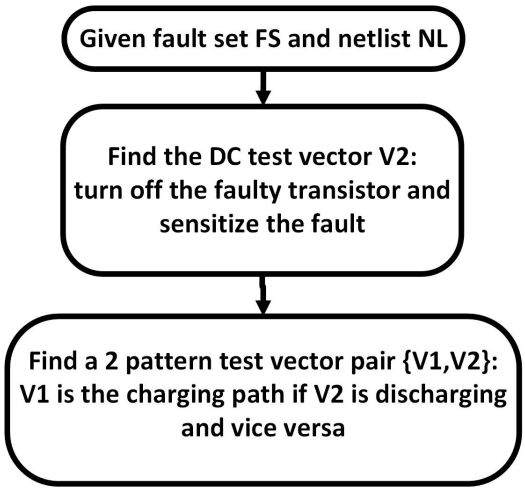


Figure 4.10: Proposed test generation flow for DS short-circuit faults

A basic flow diagram for the test pattern generation, is shown in Figure 4.10. The input to the algorithm is the fault universe FS and the netlist NL using which the algorithm determines an input vector pair $\{V_1, V_2\}$ to maximize the charging or discharging time of COUT by application of $V_2$ after $V_1$, depending on the contention caused by the defect, on application of $V_2$ due to the formation of a short-circuit path between VDD and Gnd. $V_1$ is selected to compliment the behavior of $V_2$. Each of these steps is elaborated in detail using the pseudo-code shown Algorithm 2.

According to Algorithm 2, for each defect from the fault universe FS, we find an input test vector to switch OFF the transistors in parallel to the faulty transistor, if any, using the graph G. We also, switch OFF the faulty transistor to activate the short-circuit fault. Next, for worst-case charging or discharging Elmore delay, if the defect is in pull-down network, the input vector $V_2$ must charge logic-level output capacitance (COUT) with maximum possible contention due to the faulty path in the pull-down network of G which pulls down COUT to Gnd, when the switched ON pull-up network attempts to pull up COUT to VDD. To complement input vector $V_2$ and not activate the fault, input vector $V_1$ must discharge

```
stimulus_generation(NL, FS)
Create graph G for netlist NL;
for (all defects in FS) do
    Inject defect into graph G;
    Set necessary inputs to disable all edges of G parallel to the faulty edge;
    Set necessary inputs to disable the faulty edge of G;
    if defect in pull_down network then
        Find charging vector V2 such that:;
            ED2 is maximum due to contention;
        Find discharging vector V1;
    else
        Find discharging vector V2 such that:;
            ED2 is maximum due to contention;
        Find charging vector V1;
    end
    dc_pattern.append{V2};
    two_patterns.append{V1, V2};
end
```

**Algorithm 2:** Test generation algorithm for DS short-circuit fault model

COUT. Similarly, if the fault is the pull-up network, input vector $V_2$ must discharge logic-level output capacitance (COUT) with maximum possible contention due to the faulty path in the pull-up network of G which pulls up COUT to VDD, when the switched ON pull-down network attempts to pull down COUT to Gnd. To complement input vector $V_2$ and not activate the fault, input vector $V_1$ must charge COUT. If several input vectors are applicable, then we consider all.

To find a test for the resistive DS short-circuit defect R3 in the example shown in Figure 4.9, we observe there is no parallel transistor to faulty transistor N1. To activate the defect, we switch OFF the faulty transistor N1 by input A = 0. Since the defect is in the pull-down network, we determine from the list of charging vectors generated by the algorithm, an input vector $V_2$ which provides maximum contention due to the defect pulling down COUT to Gnd, when the charging path is pulling it up to VDD. Therefore, we select $V_2$ = AB = 01, as shown in Figure 4.11(b), since having only one path charging causes the contention to be more significant in comparison to AB = 00, where two paths are charging. This gives the DC pattern for detecting the R3 defect as $\{V_2\} = \{01\}$. Next, we need to

find an input vector $V_1$, which discharges COUT thus not activate the fault. Therefore, $V_1$ = AB = 11, as shown in Figure 4.11(a). This gives the two time frame pattern for detecting the R3 fault as $\{V_1, V_2\} = \{11, 01\}$.

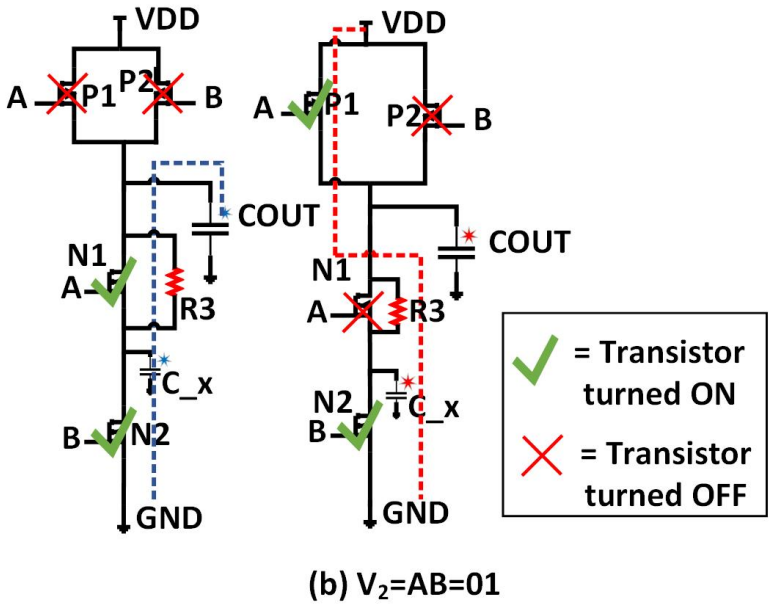


Figure 4.11: Test stimulus for R3 DS short-circuit fault in 2-input CMOS NAND gate

In case of the gates that feed an inverter (two-level gates such as AND gate, OR gate), all defects in NMOS of the inverter are sensitized, by discharging the first logic-level by applying $V_2$ and charging the first logic-level by applying $V_1$. Similarly, all defects in PMOS of the inverter are sensitized, by charging the first logic-level by applying of $V_2$ and discharging the first logic-level by applying $V_1$.

We generate a three time frame pattern test by applying all possible vector inputs before the two vectors from two time frame patterns, generated by the algorithm.

***Algorithm results***: Every fault from the fault universe for DS short-circuit faults can be detected by multiple input vector patterns, during the exhaustive analog simulation. If the pattern predicted by Algorithm 2 for a specific fault location is one of the patterns detecting the fault mapped to the same location during exhaustive simulation, then the fault is said to be detected by the algorithm. This analysis was performed to observe the coverage

of faults attained by simulating only patterns predicted by the Algorithm 2. The results obtained from this analysis are shown in Table 4.2.

Table 4.2: Fault detection for DS short-circuit faults using Algorithm 2

| Cell | M | Faults | DC | SBSI | SBDI | MBSI | MBDI | MTF |
|---|---|---|---|---|---|---|---|---|
| **AND2_X1** | 6 | 36 | 8 | 13 | 0 | 0 | 1 | 0 |
| **AOI21_X1** | 6 | 36 | 13 | 11 | 0 | 1 | 0 | 0 |
| **NAND2_X4** | 16 | 96 | 16 | 0 | 0 | 0 | 0 | 0 |
| **OR2_X2** | 8 | 48 | 10 | 12 | 0 | 2 | 0 | 2 |
| **AND3_X1** | 8 | 48 | 13 | 22 | 2 | 1 | 0 | 0 |
| **AND3_X2** | 10 | 60 | 10 | 17 | 7 | 0 | 6 | 0 |
| **AOI21_X2** | 12 | 72 | 16 | 14 | 0 | 8 | 0 | 1 |
| **NAND3_X4** | 24 | 144 | 12 | 12 | 0 | 0 | 12 | 0 |
| **OR3_X1** | 8 | 48 | 16 | 10 | 0 | 4 | 0 | 0 |
| **OR3_X2** | 10 | 60 | 12 | 16 | 0 | 0 | 8 | 0 |

In Table 4.2, column 1 shows the cell names of a few cells from the standard cell library for which algorithm coverage analysis, was performed. Column 2 shows the number of MOSFETs present in parasitic extracted netlist of each cell shown in column 1. As discussed in Section 3.1, for a cell with M MOSFETs, there will be 6M faults in fault universe for DS short-circuit fault model, which are shown in column 3. The faults detected by DC patterns generated by the algorithm are shown in column 4. Column 5 shows the faults that escaped DC testing but were detected using algorithm-generated SBSI patterns. Column 6 shows the faults that escaped DC and SBSI testing but were detected using algorithm-generated SBDI patterns. Similarly, column 7 shows the faults that escaped DC, SBSI, and SBDI testing but were detected using algorithm-generated MBSI patterns and column 8 shows the faults that escaped DC, SBSI, SBDI, and MBSI testing but were detected using algorithm-generated MBDI patterns. Column 9 shows the faults that escaped DC and all four categories of two time frame pattern (SBSI, SBDI, MBSI, and MBDI) but were detected by algorithm-generated three time frame pattern (MTF).

### 4.2.2 Algorithm for GD-GS short-circuit faults

For GD-GS short-circuit fault model, we consider resistive short defects R0-R7, between the gate and the drain terminals and between the gate and the source terminals of each transistor in a cell, as shown in Figure 4.12. This maps to 2 defects corresponding to each edge of the graph, defining the fault universe FS under consideration, with only a single defect active at a time.

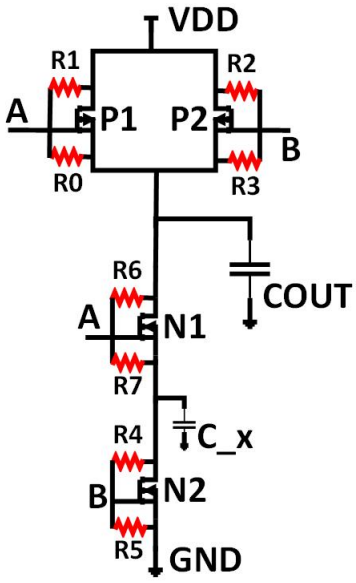


Figure 4.12: GD-GS short-circuit fault injection in 2-input CMOS NAND gate

Each input vector $V_{in}$ is categorized as either a charging vector (if it charges the logic-level output capacitance, COUT to VDD) or a discharging vector (if it discharges the logic-level output capacitance, COUT to Gnd) under the fault-free scenario. Also, an Elmore delay $ED_{in}$ consistent with the charging or the discharging time of COUT is associated with each input vector $V_{in}$.

A basic flow diagram for the test pattern generation is shown in Figure 4.13. The input to the algorithm is the fault universe FS and the netlist NL using which the algorithm determines an input vector pair $\{V_1, V_2\}$ to maximizes the charging or discharging time of COUT by application of $V_2$ after $V_1$, depending on the contention caused by the defect,

on application of $V_2$ due to the formation of a short-circuit path between VDD and Gnd (through the defect application of input at the faulty MOSFET). $V_1$ is selected to compliment the behavior of $V_2$. Each of these steps is elaborated in detail using the pseudo-code shown Algorithm 3.

According to Algorithm 3, for each defect from the fault universe FS, we find the input test vector to allow a path to be created from the defect to the logic-level output (COUT), using graph G so that the short-circuit fault can be activated. Next, for the worst-case charging or discharging Elmore delay, if the defect is in pull-down network, we determine input vector $V_2$ by switching the faulty transistor OFF (input = 0) and finding a charging vector providing maximum possible contention due to the defect in pull-down network of G, which pulls down COUT to Gnd through the input at faulty gate, when the switched ON pull-up network attempts to pull up COUT to VDD. To complement input vector $V_2$, input vector $V_1$ must discharge COUT. If we are unable to find such an input vector $V_2$, then we can also create contention by switching ON the faulty transistor (input = 1) and finding a discharging vector $V_2$ providing maximum possible contention due to the defect in pull-down network of G, which pulls up COUT to VDD through the input at faulty gate, when the switched ON pull-down network is trying to pull down COUT to Gnd. In this case, to complement input vector $V_2$, input vector $V_1$ must charge COUT. Similarly, if the defect is in pull-up network, we determine input vector $V_2$ by switching the faulty transistor ON (input = 0) and finding a charging vector providing maximum possible contention due to the defect in pull-up network of G, which pulls down COUT to Gnd through the input at faulty gate, when the switched ON pull-up network attempts to pull up COUT to VDD. To complement input vector $V_2$, input vector $V_1$ must discharge COUT. If we are unable to find such an input vector $V_2$, then we can also create contention by switching OFF the faulty transistor (input = 1) and finding a discharging vector providing maximum possible contention due to the fault in the pull-up network of G, which pulls up COUT to VDD through the input at of faulty gate, when the switched ON pull-down network is trying to

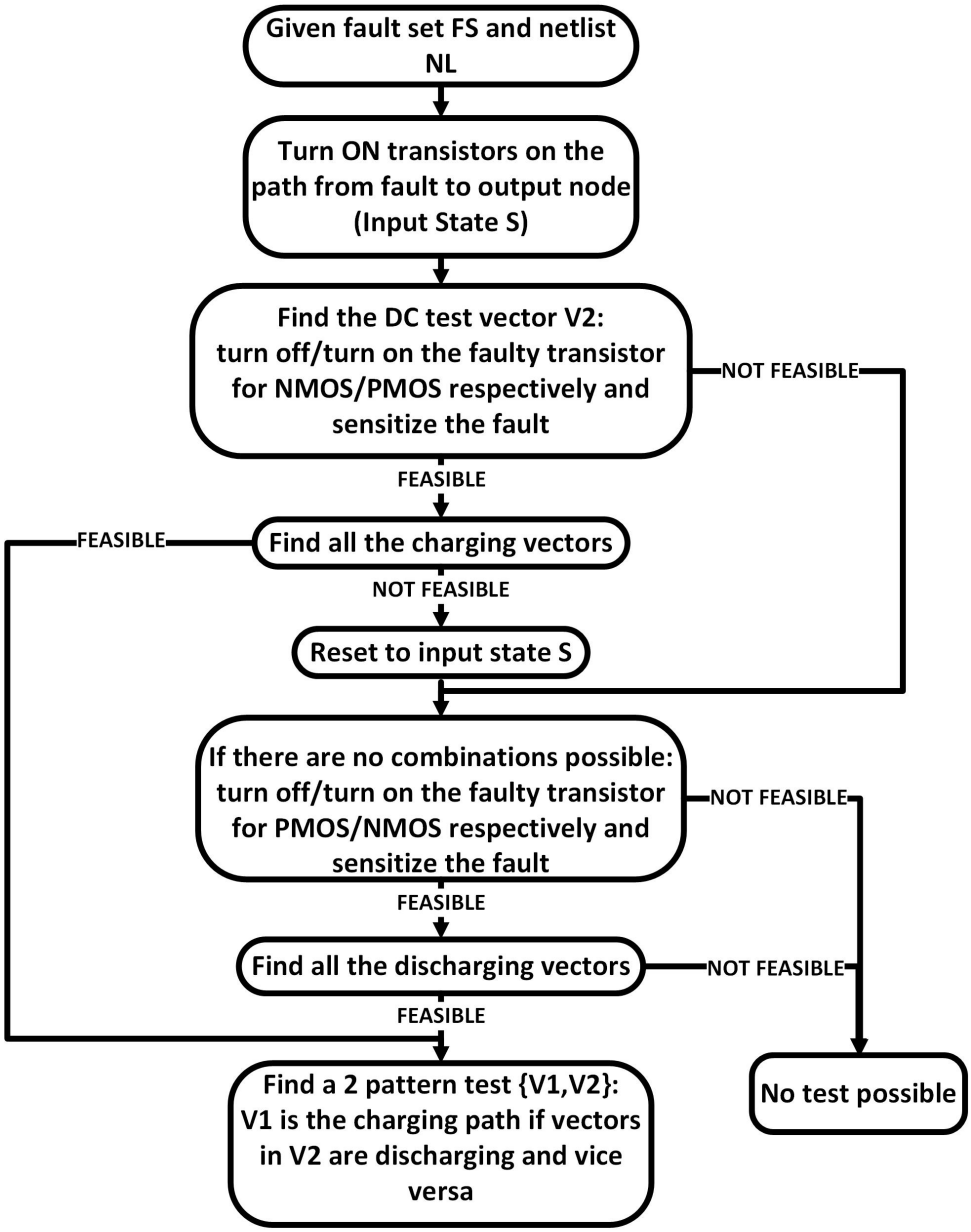


Figure 4.13: Proposed test generation flow for GD-GS short-circuit faults

```
stimulus_generation(NL, FS)
Create graph G for netlist NL;
for (all defects in FS) do
    Inject defect into graph G;
    Set necessary inputs to enable a path between the defect and the output of G;
    Save the input state S;
    if defect in pull_down network then
        if feasible, set necessary inputs to disable the faulty edge of G then
            Find charging vector V2 such that:;
               ED2 is maximum due to contention;
            if V2 is not found then
                Reset to input state S;
                Set necessary inputs to enable the faulty edge of G;
                Find discharging vector V2 such that:;
                   ED2 is maximum due to contention;
            end
        else
            Reset to input state S;
            Set necessary inputs to enable the faulty edge of G;
            Find discharging vector V2 such that:;
               ED2 is maximum due to contention;
        end
    else
        if feasible, set necessary inputs to enable the faulty edge of G then
            Find charging vector V2 such that:;
               ED2 is maximum due to contention;
            if V2 is not found then
                Reset to input state S;
                Set necessary inputs to disable the faulty edge of G;
                Find discharging vector V2 such that:;
                   ED2 is maximum due to contention;
            end
        else
            Reset to input state S;
            Set necessary inputs to disable the faulty edge of G;
            Find discharging vector V2 such that:;
               ED2 is maximum due to contention;
        end
    end
    if V2 is charging vector then
        V1 is discharging vector;
    else
        V1 is charging vector;
    end
    dc_pattern.append{V2};
    two_patterns.append{V1, V2};
end
```

**Algorithm 3:** Test generation algorithm for GD-GS short-circuit fault model

pull down COUT to Gnd. In this case, to complement input vector $V_2$, input vector $V_1$ must charge COUT. If several input vectors are applicable, then we consider all.

To find a test for the resistive short-circuit defect R4 in the example shown in Figure 4.12, we have to enable a path between the defect and COUT by switching ON transistor N1 by applying input A = 1. Since the defect is in the pull-down network, we switch OFF the faulty transistor by applying input B = 0 and find a charging vector from a list of charging vectors generated by the algorithm. Therefore, input vector AB = 10, which provides maximum contention due to the defect pulling down COUT to Gnd through the input at the gate terminal of faulty transistor N2, when the switched ON charging path is pulling it up to VDD, as shown in Figure 4.14(b). This gives the DC pattern for detecting the R4 defect as $\{V_2\} = \{10\}$. Next, we need to find an input vector $V_1$, which discharges COUT. Therefore, $V_1$ = AB = 11, as shown in Figure 4.14(a). This gives the two time frame pattern for detecting the R3 fault as $\{V_1, V_2\} = \{11, 10\}$.

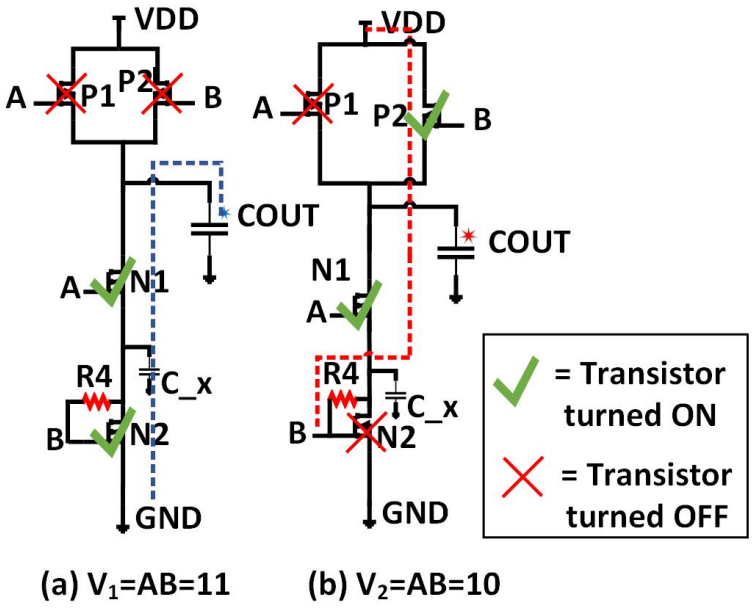


Figure 4.14: Test stimulus for R4 GD-GS short-circuit fault in 2-input CMOS NAND gate

In case of the gates that feed an inverter (two-level gates such as AND gate, OR gate), all defects in NMOS of the inverter are sensitized, by charging the first logic-level by

applying $V_2$ and discharging the first logic-level by applying $V_1$. Similarly, all defects in PMOS of the inverter are sensitized, by discharging the first logic-level by applying of $V_2$ and charging the first logic-level by applying $V_1$.

We generate a three time frame pattern test by applying all possible vector inputs before the two vectors from two time frame patterns, generated by the algorithm. It should be noted that the shorts between the gate terminal of a transistor and VDD or Gnd, will not be detected by DC tests but will be detected by two or more time frame tests.

***Algorithm results***: Every fault from the fault universe for GD-GS short-circuit faults can be detected by multiple patterns, during the exhaustive analog simulation. If the pattern predicted by Algorithm 3 for a specific fault location is one of the patterns detecting the fault mapped to the same location during exhaustive simulation, then the fault is said to be detected by the algorithm. This analysis was performed to observe the coverage of faults attained by simulating only patterns predicted by the Algorithm 3. The results obtained from this analysis are shown in Table 4.3.

Table 4.3: Fault detection for GD-GS short-circuit faults using Algorithm 3

| Cell | M | Faults | DC | SBSI | SBDI | MBSI | MBDI | MTF |
|---|---|---|---|---|---|---|---|---|
| **AND2_X1** | 6 | 72 | 12 | 29 | 3 | 0 | 1 | 0 |
| **AOI21_X1** | 6 | 72 | 32 | 17 | 0 | 1 | 0 | 1 |
| **NAND2_X4** | 16 | 192 | 20 | 4 | 0 | 0 | 0 | 0 |
| **OR2_X2** | 8 | 96 | 22 | 24 | 0 | 2 | 5 | 5 |
| **AND3_X1** | 8 | 96 | 16 | 38 | 8 | 0 | 2 | 1 |
| **AND3_X2** | 10 | 120 | 17 | 34 | 9 | 3 | 0 | 3 |
| **AOI21_X2** | 12 | 144 | 36 | 24 | 0 | 16 | 5 | 1 |
| **NAND3_X4** | 24 | 288 | 16 | 20 | 0 | 0 | 0 | 4 |
| **OR3_X1** | 8 | 96 | 36 | 31 | 0 | 1 | 1 | 2 |
| **OR3_X2** | 10 | 120 | 28 | 46 | 0 | 3 | 2 | 22 |

In Table 4.3, column 1 shows the cell names of a few cells from the standard cell library for which algorithm coverage analysis, was performed. Column 2 shows the number of MOSFETs present in parasitic extracted netlist of each cell shown in column 1. As discussed in Section 3.1, for a cell with M MOSFETs, there will be 12M faults in fault

universe for GD-GS short-circuit fault model, which are shown in column 3. The faults detected by DC patterns generated by the algorithm are shown in column 4. Column 5 shows the faults that escaped DC testing but were detected using algorithm-generated SBSI patterns. Column 6 shows the faults that escaped DC and SBSI testing but were detected using algorithm-generated SBDI patterns. Similarly, column 7 shows the faults that escaped DC, SBSI, and SBDI testing but were detected using algorithm-generated MBSI patterns and column 8 shows the faults that escaped DC, SBSI, SBDI, and MBSI testing but were detected using algorithm-generated MBDI patterns. Column 9 shows the faults that escaped DC and all four categories of two time frame pattern (SBSI, SBDI, MBSI, and MBDI) but were detected by algorithm-generated three time frame pattern (MTF).

# CHAPTER 5

# FAULT COVERAGE AND PATTERN COUNT ANALYSIS

In this chapter, we discuss and compare the number of patterns generated by the proposed algorithms in Chapter 4 with the number patterns simulated for exhaustive circuit simulations in Chapter 3, for intra-cell open and short circuit defects detection, thereby comparing the time complexities of both the methodologies for each fault model. We also compare the fault coverage obtained by the algorithms with the fault coverage obtained by the exhaustive circuit simulations. This analysis is performed to understand the trade-offs involved with both approaches.

## 5.1 Pattern count analysis

For each fault model, we compare the number of targeted patterns generated by the proposed algorithms with the number of simulated during exhaustive circuit simulations and give a factor improvement provided by the reduced number of test patterns predicted by algorithms over exhaustive simulation patterns.

### 5.1.1 Open-circuit faults

For two time frame (2TF) delay testing of a cell with 'n' input using the open-circuit fault model, the exhaustive simulation for each fault from the fault model will require the application of $2^n \times 2^n$ input stimulus patterns with $2^n$ initial conditions. Similarly, for three time frame (3TF) delay testing of such a cell will require the application of $2^n \times 2^n \times 2^n$ input stimulus patterns with $2^n$ initial conditions, as shown in Table 5.1. The two time frame and three time frame patterns, generated by the proposed Algorithm 1 are also, initialized by $2^n$ initial conditions.

Table 5.1 compares the number of patterns to be simulated for exhaustive simulation

Table 5.1: Pattern comparison for open-circuit faults

| | Ex. Simulation | | Algorithm 1 | | Improvement | |
|---|---|---|---|---|---|---|
| **Cell** | **2TF** | **3TF** | **2TF** | **3TF** | **2TF** | **3TF** |
| **OR2_X1** | 64 | 256 | 20 | 80 | 3.2 | 3.2 |
| **OR2_X2** | 64 | 256 | 20 | 80 | 3.2 | 3.2 |
| **OR2_X4** | 64 | 256 | 20 | 80 | 3.2 | 3.2 |
| **OR3_X2** | 512 | 4096 | 48 | 384 | 10.67 | 10.67 |
| **NOR3_X2** | 512 | 4096 | 48 | 384 | 10.67 | 10.67 |
| **AND3_X1** | 512 | 4096 | 48 | 384 | 10.67 | 10.67 |
| **AND3_X2** | 512 | 4096 | 48 | 384 | 10.67 | 10.67 |
| **AND3_X4** | 512 | 4096 | 48 | 384 | 10.67 | 10.67 |
| **AND2_X4** | 64 | 256 | 20 | 80 | 3.2 | 3.2 |
| **NAND2_X4** | 64 | 256 | 20 | 80 | 3.2 | 3.2 |

with the number of patterns predicted by the Algorithm 1, for both 2TF and 3TF testing using open-circuit fault model. Column 1 shows the cell names of a few cells from the standard cell library for which this analysis was performed. Columns 2 and 3 show the number of patterns that we will have to simulate in absence of proposed algorithm, for fault detection using 2TF and 3TF exhaustive circuit simulations, respectively. Columns 4 and 5 show the number of patterns generated by the proposed Algorithm 1 for 2TF and 3TF testing, respectively, using the open-circuit fault model. Column 6 and 7 show the factor improvement provided by the algorithm by reducing the number of patterns to be simulated for 2TF and 3TF testing, respectively. We can observe a significant reduction in number of patterns required to be simulated by using the algorithm.

### 5.1.2 Short-circuit faults

We performed the pattern comparison analysis comparing the pattern simulation requirement using the algorithm to generate test patterns and using the exhaustive circuit simulations, individually for each short-circuit fault model.

*DS short-circuit faults*

For DC testing of a cell with 'n' input and using DS short-circuit fault model, the exhaustive simulation for each fault from the fault model will require the application of $2^n$ input stimulus patterns. Similarly, for two time frame (2TF) delay testing of such a cell, it will require the application of $2^n \times 2^n$ input stimulus patterns for each fault. If the cell has 'M' MOSFETs, according to DS short-circuit fault model, it will have M fault locations, therefore, $2^n\times$ M DC patterns and $2^n \times (2^n - 1)\times$ M 2TF patterns will be simulated, as shown in Table 5.2.

Table 5.2: Pattern comparison for DS short-circuit faults

| | Ex. Simulation | | Algorithm 2 | | Improvement | |
|---|---|---|---|---|---|---|
| **Cell** | **DC** | **2TF** | **DC** | **2TF** | **DC** | **2TF** |
| **AND2_X1** | 24 | 72 | 8 | 14 | 3.0 | 5.1 |
| **AOI21_X1** | 48 | 336 | 10 | 40 | 4.8 | 8.4 |
| **NAND2_X4** | 64 | 192 | 16 | 32 | 4.0 | 6.0 |
| **OR2_X2** | 32 | 96 | 12 | 20 | 2.7 | 4.8 |
| **AND3_X1** | 64 | 448 | 14 | 38 | 4.6 | 11.8 |
| **AND3_X2** | 80 | 560 | 22 | 52 | 3.6 | 10.8 |
| **AOI21_X2** | 96 | 672 | 20 | 92 | 4.8 | 7.3 |
| **NAND3_X4** | 192 | 1344 | 24 | 96 | 8.0 | 14.0 |
| **OR3_X1** | 64 | 448 | 14 | 38 | 4.6 | 11.8 |
| **OR3_X2** | 80 | 560 | 22 | 52 | 3.6 | 10.8 |

Table 5.2 compares the number of patterns to be simulated for exhaustive simulation with the number of patterns predicted by the Algorithm 2 for both DC and 2TF testing, using DS short-circuit fault model. Column 1 shows the cell names of a few cells from the standard cell library for which this analysis was performed. Columns 2 and 3 show the number of patterns that we will have to simulate in absence of proposed algorithm, for fault detection using DC and 2TF exhaustive circuit simulations, respectively. Columns 4 and 5 show the number of patterns generated by the proposed Algorithm 2 for DC and 2TF testing, respectively, using the DS short-circuit fault model. The number of patterns produced by the algorithm for a cell is calculated by, summing up the patterns generated by

the algorithm for each fault location ($\sum_{i=1}^{M} P_i$, where $P_i$ is the number of patterns generated by the algorithm for a fault location i), for both DC and 2TF test. Column 6 and 7 show the factor improvement provided by the algorithm by reducing the number of patterns to be simulated for DC and 2TF testing, respectively. We can observe a significant reduction in number of patterns required to be simulated by using the algorithm.

*GD-GS short-circuit faults*

For DC testing of a cell with 'n' input and using GD-GS short-circuit fault model, the exhaustive simulation for each fault from the fault model will require the application of $2^n$ input stimulus patterns. Similarly, for two time frame (2TF) delay testing of such a cell, it will require the application of $2^n \times 2^n$ input stimulus patterns for each fault. If the cell has 'M' MOSFETs, according to GD-GS short-circuit fault model, it will have 2M fault locations, therefore, $2^n\times$ 2M DC patterns and $2^n \times (2^n - 1)\times$ 2M 2TF patterns will be simulated, as shown in Table 5.3.

Table 5.3: Pattern comparison for GD-GS short-circuit faults

| | Ex. Simulation | | Algorithm 3 | | Improvement | |
|---|---|---|---|---|---|---|
| **Cell** | **DC** | **2TF** | **DC** | **2TF** | **DC** | **2TF** |
| **AND2_X1** | 48 | 144 | 16 | 26 | 3.0 | 5.5 |
| **AOI21_X1** | 96 | 672 | 20 | 129 | 4.8 | 5.2 |
| **NAND2_X4** | 128 | 384 | 32 | 60 | 4.0 | 6.4 |
| **OR2_X2** | 64 | 192 | 21 | 48 | 3.0 | 4.0 |
| **AND3_X1** | 128 | 896 | 37 | 74 | 3.5 | 12.1 |
| **AND3_X2** | 160 | 1120 | 53 | 102 | 3.0 | 11.0 |
| **AOI21_X2** | 192 | 1344 | 40 | 258 | 4.8 | 5.2 |
| **NAND3_X4** | 384 | 2688 | 84 | 184 | 4.6 | 14.6 |
| **OR3_X1** | 128 | 896 | 24 | 112 | 5.3 | 8.0 |
| **OR3_X2** | 160 | 1120 | 40 | 140 | 4.0 | 8.0 |

Table 5.3 compares the number of patterns to be simulated for exhaustive simulation with the number of patterns predicted by the Algorithm 3 for both DC and 2TF testing, using GD-GS short-circuit fault model. Column 1 shows the cell names of a few cells from

the standard cell library for which this analysis was performed. Columns 2 and 3 show the number of patterns that we will have to simulate in absence of proposed algorithm, for fault detection using DC and 2TF exhaustive circuit simulations, respectively. Columns 4 and 5 show the number of patterns generated by the proposed Algorithm 3 for DC and 2TF testing, respectively, using the GD-GS short-circuit fault model. The number of patterns produced by the algorithm for a cell is calculated by, summing up the patterns generated by the algorithm for each fault location ($\sum_{i=1}^{2M} P_i$, where $P_i$ is the number of patterns generated by the algorithm for a fault location i), for both DC and 2TF test. Column 6 and 7 show the factor improvement provided by the algorithm by reducing the number of patterns to be simulated for DC and 2TF testing, respectively. We can observe a significant reduction in number of patterns required to be simulated by using the algorithm.

## 5.2 Fault coverage analysis

For each fault model, we compare the fault coverage obtained by the targeted patterns generated by the proposed algorithms as a percentage of the fault coverage obtained by the exhaustive circuit simulations.

### 5.2.1 Open-circuit faults

As we discussed, in the Algorithm results of Section 4.1 that if any one of the targeted patterns predicted by the Algorithm 1 for all the faults from the open-circuit fault model, detects an open-circuit fault during the exhaustive simulation for the fault model, then the fault is said to be detected by the algorithm. Based on this approach, the fault converge obtained by the algorithm, as shown in Table 4.1 is compared with the fault coverage obtained by exhaustive simulation, as shown in Table 3.1, for open-circuit fault model.

Table 5.4 shows the total fault coverage obtained by the exhaustive simulation versus the total fault coverage by the proposed algorithm for the open-circuit fault model. Column 1 shows the cell names of a few cells from the standard cell library, for which this

Table 5.4: Algorithm 1 vs Simulation coverage for open-circuit faults

| Cell | Simulation | Algorithm 1 | Percentage |
|---|---|---|---|
| **OR2_X1** | 120 | 115 | 95.83 |
| **OR2_X2** | 160 | 145 | 90.62 |
| **OR2_X4** | 307 | 279 | 90.88 |
| **OR3_X2** | 210 | 174 | 82.86 |
| **NOR3_X2** | 184 | 184 | 100 |
| **AND3_X1** | 163 | 151 | 92.64 |
| **AND3_X2** | 206 | 196 | 95.15 |
| **AND3_X4** | 418 | 401 | 95.93 |
| **AND2_X4** | 296 | 290 | 97.97 |
| **NAND2_X4** | 172 | 149 | 86.63 |

analysis was performed. Column 2 shows the total number of faults detected by exhaustive simulation and Column 3 shows the total number of faults detected, if only the patterns generated by the algorithm were simulated, for the open-circuit fault model. Column 4 shows the percentage of total detectable faults (through exhaustive simulation) that were also detected by the algorithm proposed patterns. From the table, we can observe that only a few faults from the fault universe under consideration, are missed by the test generated by the algorithm, which will be detected by exhaustive simulation.

### 5.2.2 Short-circuit faults

We performed the fault coverage analysis comparing the fault coverage obtained through the algorithm generated test patterns and through the exhaustive circuit simulations, individually for each short-circuit fault model.

*DS short-circuit short-circuit faults*

As we discussed, in the Algorithm results of Section 4.2.1, that if for a fault at a certain location any one of the targeted patterns predicted by the Algorithm 2 also exist in the fault set detecting the fault during the exhaustive simulation for DS short-circuit fault model, then the fault is said to be detected by the algorithm. Based on this approach, the fault con-

verge obtained by the algorithm, as shown in Table 4.2 is compared with the fault coverage obtained by exhaustive simulation, as shown in Table 3.2, for DS short-circuit fault model.

Table 5.5: Algorithm 2 vs Simulation coverage for DS short-circuit faults

| Cell | Simulation | Algorithm 2 | Percentage |
|---|---|---|---|
| **AND2_X1** | 22 | 22 | 100 |
| **AOI21_X1** | 25 | 25 | 100 |
| **NAND2_X4** | 16 | 16 | 100 |
| **OR2_X2** | 26 | 26 | 100 |
| **AND3_X1** | 38 | 38 | 100 |
| **AND3_X2** | 51 | 40 | 78.43 |
| **AOI21_X2** | 39 | 39 | 100 |
| **NAND3_X4** | 36 | 36 | 100 |
| **OR3_X1** | 30 | 30 | 100 |
| **OR3_X2** | 45 | 36 | 80 |

Table 5.5 shows the total fault coverage obtained by the exhaustive simulation versus the total fault coverage by the proposed algorithm for DS short-circuit fault model. Column 1 shows the cell names of a few cells from the standard cell library, for which this analysis was performed. Column 2 shows the total number of faults detected by exhaustive simulation and Column 3 shows the total number of faults detected, if only the patterns generated by the algorithm were simulated, for the DS short-circuit fault model. Column 4 shows the percentage of total detectable faults (through exhaustive simulation) that were also detected by the algorithm proposed patterns. From the table, we can observe that for a few cells, only a few faults from the fault universe under consideration, are missed by the test generated using the algorithm, for each fault location, which will be detected by exhaustive simulation.

*GD-GS short-circuit faults*

As we discussed, in the Algorithm results of Section 4.2.2, that if for a fault at a certain location any one of the targeted patterns predicted by the Algorithm 3 also exist in the fault set detecting the fault during the exhaustive simulation for GD-GS short-circuit fault

model, then the fault is said to be detected by the algorithm. Based on this approach, the fault converge obtained by the algorithm, as shown in Table 4.3 is compared with the fault coverage obtained by exhaustive simulation, as shown in Table 3.3, for GD-GS short-circuit fault model.

Table 5.6: Algorithm 3 vs Simulation coverage for GD-GS short-circuit faults

| Cell | Simulation | Algorithm 3 | Percentage |
|---|---|---|---|
| **AND2_X1** | 46 | 45 | 97.83 |
| **AOI21_X1** | 51 | 51 | 100 |
| **NAND2_X4** | 24 | 24 | 100 |
| **OR2_X2** | 62 | 58 | 93.55 |
| **AND3_X1** | 72 | 65 | 90.28 |
| **AND3_X2** | 91 | 66 | 72.53 |
| **AOI21_X2** | 82 | 82 | 100 |
| **NAND3_X4** | 52 | 40 | 76.92 |
| **OR3_X1** | 73 | 71 | 97.26 |
| **OR3_X2** | 103 | 101 | 98.06 |

Table 5.6 shows the total fault coverage obtained by the exhaustive simulation versus the total fault coverage by the proposed algorithm for GD-GS short-circuit fault model. Column 1 shows the cell names of a few cells from the standard cell library, for which this analysis was performed. Column 2 shows the total number of faults detected by exhaustive simulation and Column 3 shows the total number of faults detected, if only the patterns generated by the algorithm were simulated, for the GD-GS short-circuit fault model. Column 4 shows the percentage of total detectable faults (through exhaustive simulation) that were also detected by the algorithm proposed patterns. From the table, we can observe that only a few faults from the fault universe under consideration, are missed by the test generated using the algorithm, for each fault location, which will be detected by exhaustive simulation.

# CHAPTER 6
# CONCLUSION

In this thesis, we demonstrate that the current methodology for structural testing of open and short-circuit defects within the standard cells used as building blocks of all Integrated Circuit (IC) chips is insufficient. There exist certain magnitudes of open and short-circuit faults that can go undetected by the current methodology since they only assume the extreme value of resistance for these defects and only use up to two time frame testing with limited number of test patterns (mostly with single bit change in the two vectors of two time frame pattern). It is demonstrated through exhaustive simulations, that defects escaping current testing methodology can be detected by the use of multi-bit change and multi time frame input patterns as a stimulus applied to the cell.

We also proposed a systematic methodology for test pattern generation for resistive open and short-circuit defects within the standard cell. We presented algorithms based on the Elmore-delay analysis and short-circuit analysis using the switch-level transistor model, to identify input pattern stimuli to detect intra-cell open and short-circuit defects. The use of patterns generated by the algorithms for guided simulations, allows us to achieve orders of magnitude of speed-up over exhaustive simulation approach with a small loss in fault coverage from exhaustive simulation for up to three time frame pattern test.

Future work will focus on implementing and observing the correlation of the extension of the open-circuit algorithm for mirror logic implementation of cells with exhaustive simulation results for the same. This works also, lays out the stepping stones for the future development of a fully automated, circuit-level, SCAN test pattern generation algorithm including complete fault models for open and short circuit intra-cell defects, for effective fault detection.

## REFERENCES


[1] K. Y. Cho, S. Mitra, and E. J. McCluskey, "Gate exhaustive testing," in *IEEE International Conference on Test, 2005.*, 2005, 7 pp.–777.

[2] S. K. Goel, K. Chakrabarty, M. Yilmaz, K. Peng, and M. Tehranipoor, "Circuit Topology-Based Test Pattern Generation for Small-Delay Defects," in *2010 19th IEEE Asian Test Symposium*, 2010, pp. 307–312.

[3] P. Banerjee and J. A. Abraham, "Characterization and Testing of Physical Failures in MOS Logic Circuits," *IEEE Design Test of Computers*, vol. 1, no. 3, pp. 76–86, 1984.

[4] F. Hapke and J. Schloeffel, "Introduction to the defect-oriented cell-aware test methodology for significant reduction of DPPM rates," in *2012 17th IEEE European Test Symposium (ETS)*, 2012, pp. 1–6.

[5] F. Hapke, W. Redemund, A. Glowatz, J. Rajski, M. Reese, M. Hustava, M. Keim, J. Schloeffel, and A. Fast, "Cell-Aware Test," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 33, no. 9, pp. 1396–1409, 2014.

[6] A. D. Singh, "Cell Aware and stuck-open tests," in *2016 21th IEEE European Test Symposium (ETS)*, 2016, pp. 1–6.

[7] S. P. Dixit, D. D. Vora, and K. Peng, "Challenges in Cell-Aware Test," in *2018 IEEE 23rd European Test Symposium (ETS)*, 2018, pp. 1–6.

[8] X. Lin and S. M. Reddy, "On generating high quality tests based on cell functions," in *2015 IEEE International Test Conference (ITC)*, 2015, pp. 1–9.

[9] K. Fuchs, H. C. Wittmann, and K. J. Antreich, "Fast test pattern generation for all path delay faults considering various test classes," in *Proceedings ETC 93 Third European Test Conference*, 1993, pp. 89–98.

[10] K. Charafeddine and F. Ouardi, "Fast timing characterization of cells in standard cell library design based on curve fitting," in *2017 International Conference on Wireless Technologies, Embedded and Intelligent Systems (WITS)*, 2017, pp. 1–6.

[11] K. C. Y. Mei, "Bridging and stuck-at faults," *IEEE Transactions on Computers*, vol. C-23, no. 7, pp. 720–727, 1974.

[12] Y. Kung, K. Lee, and S. M. Reddy, "Generating compact test patterns for stuck-at faults and transition faults in one atpg run," in *2018 IEEE International Test Conference in Asia (ITC-Asia)*, 2018, pp. 1–6.

[13] D. Arumi, R. Rodriguez-Montanes, and J. Figueras, "Experimental characterization of cmos interconnect open defects," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 27, no. 1, pp. 123–136, 2008.

[14] H. Cox and J. Rajski, "Stuck-open and transition fault testing in cmos complex gates," in *International Test Conference 1988 Proceeding@m_New Frontiers in Testing*, 1988, pp. 688–694.

[15] I. Pomeranz and S. M. Reddy, "On the number of tests to detect all path delay faults in combinational logic circuits," *IEEE Trans. Comput.*, vol. 45, no. 1, pp. 50–62, Jan. 1996.

[16] X. Lin, K. Tsai, C. Wang, M. Kassab, J. Rajski, T. Kobayashi, R. Klingenberg, Y. Sato, S. Hamada, and T. Aikyo, "Timing-aware atpg for high quality at-speed testing of small delay defects," in *2006 15th Asian Test Symposium*, 2006, pp. 139–146.

[17] A. Sinha, S. Pandey, A. Singhal, A. Sanyal, and A. Schmaltz, "Dfm-aware fault model and atpg for intra-cell and inter-cell defects," in *2017 IEEE International Test Conference (ITC)*, 2017, pp. 1–10.

[18] I. Pomeranz, "Multi-pattern $n$-detection stuck-at test sets for delay defect coverage," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 20, no. 6, pp. 1156–1160, 2012.

[19] J. Geuzebroek, E. J. Marinissen, A. Majhi, A. Glowatz, and F. Hapke, "Embedded multi-detect atpg and its effect on the detection of unmodeled defects," in *2007 IEEE International Test Conference*, 2007, pp. 1–10.

[20] A. Jas, S. Natarajan, and S. Patil, "The Region-Exhaustive Fault Model," in *Proc. of Asian Test Symposium*, 2007, pp. 13–18.

[21] A. Jee and F. J. Ferguson, "An analysis of shorts in cmos standard cell circuits," in *Proceedings Seventh Annual IEEE International ASIC Conference and Exhibit*, 1994, pp. 362–365.

[22] A. Chehab, A. Kayssi, and A. Ghandour, "Transient Current Testing of Gate-Oxide Shorts in CMOS," in *2007 2nd International Design and Test Workshop*, 2007, pp. 77–81.

[23] S. Natarajan, S. K. Gupta, and M. A. Breuer, "Switch-level delay test," in *Proc. of International Test Conference*, 1999, pp. 171–180.

[24] Galiay, Crouzet, and Vergniault, "Physical versus logical fault models mos lsi circuits: Impact on their testability," *IEEE Transactions on Computers*, vol. C-29, no. 6, pp. 527–531, 1980.

[25] P. Engelke, I. Polian, H. Manhaeve, M. Renovell, and B. Becker, "Delta-IDDQ Testing of Resistive Short Defects," in *2006 15th Asian Test Symposium*, 2006, pp. 63–68.

[26] R. Guo, B. Archer, K. Chau, and X. Cai, "Efficient cell-aware defect characterization for multi-bit cells," in *2018 IEEE International Test Conference in Asia (ITC-Asia)*, 2018, pp. 7–12.

[27] F. Hapke, M. Reese, J. Rivers, A. Over, V. Ravikumar, W. Redemund, A. Glowatz, J. Schloeffel, and J. Rajski, "Cell-aware production test results from a 32-nm notebook processor," in *2012 IEEE International Test Conference*, 2012, pp. 1–9.

[28] A. Prabhu, V. Vorisek, H. Lang, and T. Schumann, "Analysis of cell-aware test pattern effectiveness — a case study using a 32-bit automotive microcontroller," in *2014 19th IEEE European Test Symposium (ETS)*, 2014, pp. 1–2.

[29] F. Hapke, J. Schloeffel, W. Redemund, A. Glowatz, J. Rajski, M. Reese, J. Rearick, and J. Rivers, "Cell-aware analysis for small-delay effects and production test results from different fault models," in *2011 IEEE International Test Conference*, 2011, pp. 1–8.

[30] W. Howell, F. Hapke, E. Brazil, S. Venkataraman, R. Datta, A. Glowatz, W. Redemund, J. Schmerberg, A. Fast, and J. Rajski, "Dppm reduction methods and new defect oriented test methods applied to advanced finfet technologies," in *2018 IEEE International Test Conference (ITC)*, 2018, pp. 1–10.

[31] "NanGate FreePDK45 Generic Open Cell Library Si2."

[32] P. Franco and E. J. McCluskey, "Three-pattern tests for delay faults," in *Proceedings of IEEE VLSI Test Symposium*, 1994, pp. 452–456.